Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Localizzazione e riconoscimento di eventi sonori mediante array microfonico

Tesi di Laurea Magistrale in Ingegneria Informatica

Laureando:
Davide Pagin

Relatore:
Antonio Rodà

Anno Accademico 2015/2016

Sommario

In contesti di video-sorveglianza o di tele-assistenza i suoni prodotti nell'ambiente osservato possono contenere informazioni utili a rilevare in maniera automatica potenziali eventi di interesse, ad esempio situazioni di pericolo.
Questa tesi descrive una implementazione di localizzazione e riconoscimento automatico di eventi sonori normalmente associati a situazioni di pericolo (urla e colpi di arma da fuoco), utilizzando un array microfonico integrato all'interno di un sensore di basso costo e larga diffusione, il Kinect v1 di Microsoft.
Nel caso della localizzazione, sono stati implementati due algoritmi: il primo permette di ottenere le registrazioni compiute dall'array di microfoni del Kinect; il secondo stima tramite cross-correlation i valori di TDOA tra i vari microfoni, utilizzati successivamente per il calcolo dell'angolazione della sorgente sonora.
Nel caso del riconoscimento sonoro, è stato creato un dataset tramite la raccolta di 120 campioni audio d'interesse. Successivamente sono state estratte determinate feature audio che sono state analizzate da vari modelli di Machine Learning (bayesiani, alberi, ensemble methods...) per determinare la classe di appartenenza dei vari campioni (urla, arma da fuoco, suono random).
I risultati della localizzazione hanno mostrato una stima dell'angolazione con errori medi di circa 3°- 4° per le urla e di circa 2° per i colpi da sparo. Inoltre la presenza di rumore di sottofondo non compromette il calcolo dell'angolazione se non a livelli SNR prossimi a 0.
I test sul riconoscimento sonoro hanno evidenziato come il modello con le migliori prestazioni di predizione sia stato il Random Forest, con valori di correttezza del 96-97% con entrambi gli eventi sonori.
I risultati ottenuti mostrano che, anche utilizzando microfoni integrati a basso costo, è possibile ottenere prestazioni paragonabili allo stato dell'arte e nel caso del riconoscimento sonoro, utilizzando come modello il Random Forest, anche superiori.

Alla mia famiglia ed ai miei amici, che hanno contribuito alla realizzazione di questo lavoro di tesi.

Indice

1 Introduzione **6**

2 Localizzazione Sonora **8**
2.1 Kinect e libreria software 8
2.1.1 Dispositivo Kinect v1 8
2.1.2 *libfreenect* 10
2.2 Angolazione sonora 12
2.2.1 TDOA 12
2.2.2 Cross-correlation 12
2.2.3 Calcolo dell'angolazione 14

3 Riconoscimento di eventi sonori **16**
3.1 Eventi sonori 16
3.2 Audio features 17
3.2.1 Zero-Crossing Rate 18
3.2.2 Centroide Spettrale 19
3.2.3 Spectral Flatness Measure 19
3.2.4 Spectral Skewness 19
3.2.5 Attack Slope e Decrease Slope 20
3.2.6 MFCC 20
3.3 Machine Learning 21
3.3.1 C4.5 21
3.3.2 Random Forest 22
3.3.3 AdaBoostM1 24
3.3.4 AODE 25
3.3.5 GMM-EM 26
3.3.6 Metodo di validazione del classificatore 28
3.4 Implementazione del riconoscimento 29

4 Risultati **32**
4.1 Localizzazione sonora 32
4.1.1 Setup dell'esperimento 32
4.1.2 Esecuzione dei test su angolazione 32
4.1.3 Quantizzazione 33
4.1.4 Risultati su angolazioni stimate 34
4.1.5 Risultati su angolazioni in presenza di AWGN 37

4.2 Riconoscimento sonoro 38
4.2.1 Setup dell'esperimento 38
4.2.2 Risultati del 10-cross-validation 39
4.2.3 Confronto con risultati simili dello stato dell'arte 42

5 Conclusioni e sviluppi futuri **44**
5.1 Conclusioni 44
5.2 Sviluppi futuri 44

Capitolo 1

Introduzione

La video-sorveglianza è sempre stato un settore molto importante nel campo dei dispositivi per ambienti privati e pubblici. Secondo il report compiuto da Seagate del 2015 [1], circa il 75% delle aziende intervistate ha risposto che il numero delle video-camere da loro installate continuerà ad aumentare così come il rispettivo valore strategico. Inoltre l'88% ha dichiarato di possedere camere in funzione 24 ore su 24. Come campi applicativi della video sorveglianza sono stati resi i risultati visibili nella Fig.1.1. I maggiori campi corrispondono al video tracking (68%), al motion detection (62%) e all'object detection (59%). Infine, sempre secondo [1], l'85% dei motivi esposti dalle aziende per l'utilizzo di tali camere comprende la sicurezza, il 61% la riduzione di furti nelle proprie strutture, e il 58% la prevenzione di atti di vandalismo.
Tuttavia l'uso di detection manuale da parte di un centro di controllo si rivela ormai piuttosto impraticabile, visto l'aumento del numero di video-camere installate negli edifici. Per questo motivo molte ricerche si sono concentrate nella video-sorveglianza automatica, senza la presenza di un utente partecipe. Uno degli ausili più utili alla video-sorveglianza consiste nell'utilizzo di sensori audio in grado di registrare l'ambiente circostante e, nel caso di riconoscimento di eventi anomali, segnalare importanti informazioni al sistema automatizzato, come per esempio la localizzazione dell'origine di tale evento. In questo modo la video-camera si può ridirezionare autonomamente, inquadrare la zona di interesse e individuare la causa dell'anomalia.
In questo elaborato verrà proposto un sistema di localizzazione spaziale, intesa come angolazione di ricezione, della sorgente sonora e di riconoscimento di due particolari eventi sonori: le urla prodotte da persone ("Scream") e il suono prodotto da colpi di arma da fuoco ("Shotgun"). Per il primo task si è utilizzato il dispositivo Microsoft Kinect v1, sviluppato da Microsoft, che dispone di sensori audio e video. In particolare, tramite l'utilizzo dell'array di microfoni installati, si attua la predizione dell'angolazione di ricezione sfruttando i TDOA percepiti dai microfoni. Per l'implementazione software della registrazione audio si è usufruito della libreria opensource *libfreenect* sviluppata dalla community OpenKinect.
Per il secondo task sono state estratte alcune feature audio di interesse su un insieme di 120 campioni contenenti suoni di scream, shotgun e suoni randomici non appartenenti ai due tipi precedenti. Sono stati successivamente analizzati vari modelli di Machine Learning (modelli bayesiani, alberi, ensemble methods, clustering supervi-

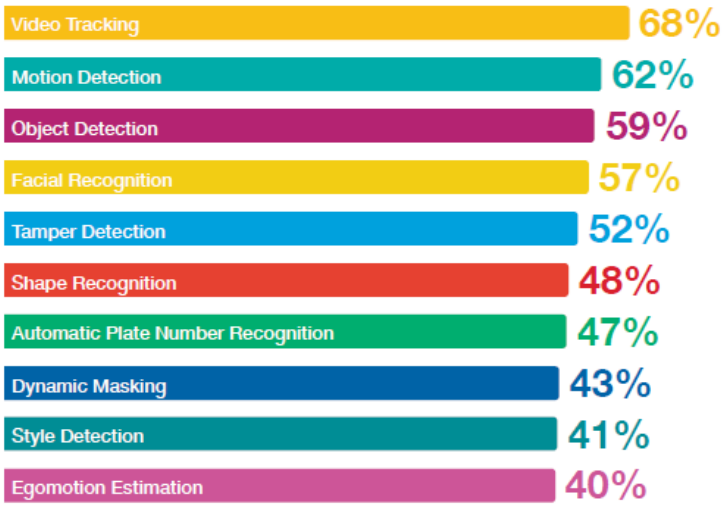


Figura 1.1: I campi applicativi delle video-camere secondo i responsi delle aziende intervistate dall'azienda Seagate [1].

sionato) e applicati su tali campioni. Dopo aver valutato i test, basati sul 10-cross validation, si è scelto di promuovere il modello Random Forest, essendo il migliore nella predizione di ciascuno dei due eventi d'interesse.
Questo elaborato è organizzato come segue:

- nel Capitolo 2 verrà descritta l'implementazione della registrazione audio via Kinect v1 e della localizzazione sonora tramite TDOA;
- nel Capitolo 3 verranno presentate le caratteristiche degli eventi sonori prescelti, le feature audio estratte, i modelli di Machine Learning utilizzati e l'implementazione del riconoscimento sonoro;
- nel Capitolo 4 saranno descritti i risultati ottenuti sia nella localizzazione della sorgente sonora, sia nel riconoscimento degli eventi sonori;
- nel Capitolo 5.1 saranno tratte le conclusioni e i possibili sviluppi futuri.

Capitolo 2

Localizzazione Sonora

Come elencato nell'Introduzione, la prima fase del progetto di tesi consiste nel localizzare l'evento sonoro, senza analizzare per il momento la tipologia di suono (questo sarà il tema del Capitolo 3). Per conseguire tale obiettivo, l'utilizzo di un solo microfono per la registrazione audio risulta inutile poiché l'analisi sonora di tale registrazione non contiene alcuna informazione riguardante la provenienza della sorgente sonora. Per questo motivo è necessario l'utilizzo di un array di microfoni [2], un insieme di microfoni disposti nello spazio di ricezione in posizioni diverse in modo da registrare la sorgente sonora in una diversa "prospettiva". Questa configurazione, oltre a permettere una migliore ricezione sonora dato l'utilizzo di più microfoni, permette anche di poter ricavare l'origine della sorgente sonora, come si vedrà nella seconda parte di questo Capitolo. In particolare, la sorgente sonora sarà registrata in diversi canali, uno per ogni microfono.
Per localizzazione si possono intendere vari task in base alle caratteristiche dell'array di microfoni utilizzato e agli obiettivi che si vogliono prefissare. Si può intendere:

- l'angolazione di ricezione;
- la posizione della sorgente sonora nel piano che comprende il dispositivo (cioè la posizione nelle coordinate (x,y);
- la posizione 3D della sorgente sonora.

Per le finalità del lavoro di tesi, l'obiettivo consiste nel ricavare l'angolazione di ricezione della sorgente sonora da parte del dispositivo munito dell'array di microfoni.

2.1 Kinect e libreria software

2.1.1 Dispositivo Kinect v1

Esistono numerosi dispositivi muniti di array di microfoni in grado di adempiere agli obiettivi prefissati. Tuttavia una delle linee guida per il progetto di tesi limitava la scelta su prodotti che non avessero un costo eccessivo. Tale considerazione ha portato a valutare, e alla fine scegliere, come dispostivo il prodotto denominato Kinect, in particolare il modello Kinect v1 [3].

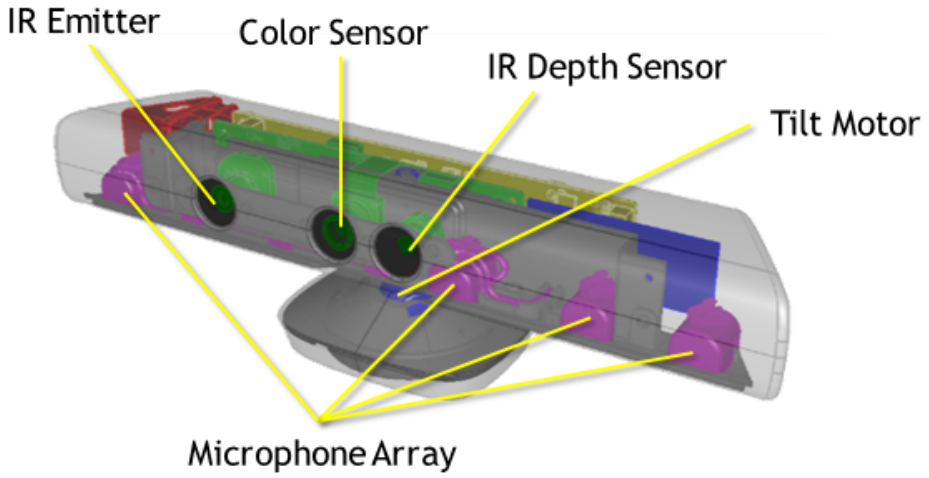


Figura 2.1: Kinect v1.

Kinect è un device sviluppato dalla società Microsoft, concepito all'inizio come accessorio per la console da gaming Xbox 360. Tale device permette di riconoscere i gesti compiuti dai videogiocatori in modo che questi possano interagire, senza l'ausilio del joypad, con determinati giochi supportati, aumentando il senso di coinvolgimento. Il Kinect inoltre è in grado di riconoscere i comandi vocali impartiti dagli utenti, come per esempio decidere quale applicazione avviare nel menu iniziale.
In tal senso, il lancio di tale prodotto è stato un successo: secondo Ndp Group, società specializzata in inchieste di mercato, Microsoft ha venduto circa 2,5 milioni di dispositivi Kinect nei primi 25 giorni, mentre la stessa Microsoft ha annunciato nei primi mesi dell'anno 2013 di aver venduto complessivamente ben 24 milioni di Kinect v1.
Successivamente al lancio, si diffusero librerie e driver per poter utilizzare il dispositivo Kinect come device per PC. Infatti, se all'inizio la fascia di mercato riguardava solamente l'ambito videoludico, l'alto rapporto qualità-prezzo ha contribuito alla sua diffusione anche in altri ambiti, come per esempio la robotica, per le feature video-audio che il Kinect dispone.
Come caratteristiche hardware, il Kinect presenta sia sensori video [4][6] che audio. Per quanto riguarda la componente video, sono presenti una telecamera RGB e un doppio sensore di profondità a raggi infrarossi, composto da un proiettore a infrarossi e da una telecamera sensibile alla medesima banda. La telecamera RGB ha una risoluzione di 640x480 pixel, mentre quella a infrarossi di 320x240 pixel, e ha un frame rate di 30 FPS. Il viewing angle risulta essere di 43° lungo la verticale e di 57° lungo l'orizzontale.
Per quanto riguarda invece la componente audio, il Kinect dispone di un array di quattro microfoni disposti linearmente all'interno del dispositivo, in cui uno dei quattro microfoni si trova ad una sua estremità mentre i rimanenti tre all'estremità opposta. Tale disposizione è giustificata dal fatto che la localizzazione sonora avviene tramite l'utilizzo dei microfoni a coppie: infatti, riferendoci alla Fig.2.1 e numerando il microfono isolato come 1 e i rimanenti come 2, 3 e 4 (da sinistra verso destra), le coppie di microfoni utilizzate risultano essere rispettivamente (1,2), (1,3), (1,4).

Kinect	Array Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	±27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

Figura 2.2: Specifiche tecniche del Kinect v1, incluse in [6].

Come verrà illustrato successivamente, per il calcolo dell'angolazione della sorgente sonora è necessario disporre di una coppia di microfoni. Avendo a disposizione non una sola coppia, bensì tre coppie di microfoni, si possono commettere meno errori nel calcolo dell'angolazione. Inoltre, tramite tecniche di beamforming [5], è possibile isolare la sorgente sonora in una stanza affetta da rumore di sottofondo. La frequenza di campionamento dei microfoni a disposizione è di 16KHz.
Nella Fig.2.3 è mostrata la geometria dell'array di microfoni, in particolare la disposizione spaziale dei microfoni, necessario per il calcolo dell'angolazione della sorgente sonora. Tale geometria limita il range d'angolo che il dispositivo Kinect può ricevere, cioè [0°, 180°], poichè la linearità dell'array di microfoni non permette di distinguere se un suono provenga frontalmente o posteriormente.

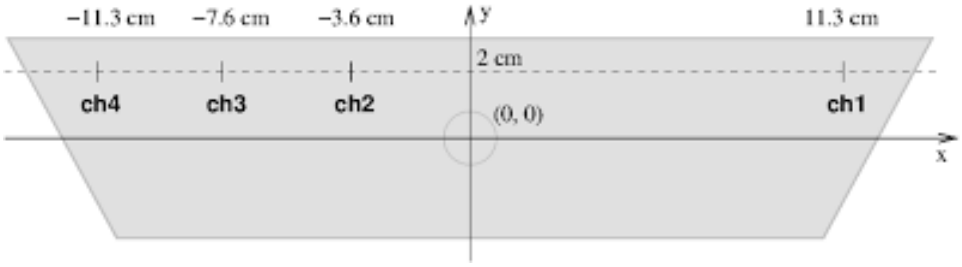


Figura 2.3: Disposizione spaziale dei 4 microfoni rispetto al baricentro del dispositivo Kinect v1.

Infine il Kinect è in grado di inclinarsi verticalmente di 27° gradi a seconda delle necessità di campo audio-visivo.

2.1.2 *libfreenect*

Scelto il dispositivo per registrare l'audio, occorre scegliere la libreria software adatta per poter effettivamente compiere tale operazione. La libreria più famosa relativa al dispositivo Kinect è l'SDK di Windows [7], che permette di sviluppare applicazioni in diversi linguaggi ed ha a disposizione un'ampia documentazione. Tuttavia tale SDK è limitato all'ambiente Windows, quindi non può essere adoperato per altri sistemi operativi. Infatti la scelta della libreria si è basata su alcuni requisiti:

1. tipologia open-source;
2. compatibilità con tutti i sistemi operativi Windows, Linux e Mac OS;
3. presenza di metodi per poter usufruire della componente audio del dispositivo Kinect.

Tali requisiti hanno portato a scegliere come libreria *libfreenect*, sviluppata da OpenKinect. OpenKinect è una community opensource composta da oltre 2000 membri che sviluppa librerie e driver appositi per poter utilizzare le funzionalità dei dispositivi Kinect su qualsiasi sistema operativo. Il loro prodotto più famoso è la libreria *libfreenect*. Essa include driver e API cross-piattaforma compatibili con Windows, Linux, e Mac OS. Tale libreria può essere utilizzata con vari linguaggi di programmazione come C, C++ e Java e comprende alcune funzionalità come l'hand tracking, lo skeleton tracking, il depth processing e, per il contesto di questa tesi, la registrazione audio con i microfoni incorporati.
Dopo aver scelto la libreria *libfreenect*, è stato sviluppato il codice in linguaggio **C** in grado di registrare l'audio dei quattro microfoni in quattro file audio in formato WAV (nell'Alg.1 è presente il pseudocodice). Il codice inizializza il device e i quattro file audio che conterranno le registrazioni audio di ciascun microfono, e associa al device una chiamata di callback. Dopo il lancio della registrazione, ogni qualvolta il dispositivo acquisisce nuovi dati audio viene chiamato il callback, che compie l'update dei file audio. Quando si decide di concludere la registrazione audio, il file header di ciascun file viene configurato in modo da convalidare il tutto.

Algorithm 1 Registrazione audio

```
procedure CALLBACK
    Scrive il nuovo contenuto audio ricevuto nei 4 file audio di output;
    Aumenta il numero di sample totali registrati;
procedure MAIN
    Verifica che il numero di dispositivi Kinect trovati sia superiore a 0;
    Apre la connessione con il Kinect;
    Associa i file audio di output al dispositivo;
    Associa la chiamata di callback al dispositivo;
    Lancia l'acquisizione dell'audio;
    ▷ La condizione di stop può essere un Timer oppure la diretta terminazione
    inviata via terminale (CTRl-C)
    while condizione di stop AND processi in corso ≥ 0 do
        do something;
    for file audio di output del microfono i do
        rende valido il file header WAV di i;
        chiude il file audio i;
    Termina il collegamento con il device Kinect;
```

2.2 Angolazione sonora

2.2.1 TDOA

Per determinare l'angolazione della sorgente sonora rispetto al ricevitore, una tecnica utilizzata consiste nel derivare i Time Difference of Arrival (**TDOA**), cioè i delay di ricezione del suono tra microfoni. Essi sono dovuti alla differenza di distanza compiuta dall'onda sonora della sorgente a raggiungere i diversi microfoni.

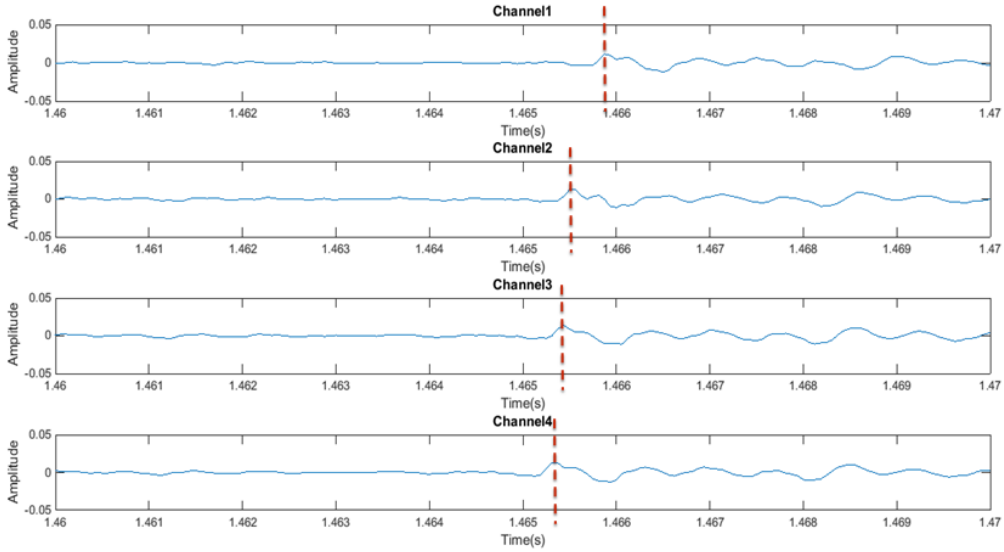


Figura 2.4: Esempio di delay tra le varie registrazioni effettuate nei vari canali (cioè i microfoni del Kinect v1).

Nella Fig.2.4 viene mostrato un esempio del TDOA nelle registrazioni di ciascun microfono del Kinect v1. Nel caso preso in esame è stato evidenziato l'inizio della ricezione della sorgente sonora da parte dei quattro microfoni, in particolare il primo picco in ampiezza rilevato. Come si può rilevare, il primo microfono a rilevare il picco risulta essere il canale (microfono) 4, seguito subito in ordine dai canali 3 e 2. Infine, più distanziato temporalmente, il canale 1 rileva il picco sonoro. Date la disposizione dell'array di microfoni usato, un suono viene sempre ricevuto prima/dopo dal microfono 1 e dopo/prima dai rimanenti microfoni, con un delay tra questi ultimi molto ridotto. Questa è una delle motivazioni per cui si utilizzano le coppie (1,2), (1,3), (1,4), poiché con questa scelta si ha a disposizione una precisione maggiore dovuto alla distanza maggiore (con conseguente TDOA maggiore) tra i microfoni di ciascuna coppia.

2.2.2 Cross-correlation

Un metodo automatico per ricavare i TDOA consiste nell'utilizzare la cross-correlation (correlazione incrociata). Dato $x_i[n]$ e $x_j[n]$ due registrazioni audio di due microfoni i e j, campionati con N samples, la cross-correlation è espressa dalla seguente formula:

$$R_{ij}(\tau) = \sum_{n=0}^{N-1} x_i[n]\, x_j[n-\tau] \tag{2.1}$$

La cross-correlation risulta essere massima quando τ corrisponde al delay tra i due segnali audio dei due microfoni. Tale calcolo può risultare oneroso poiché la complessità di tale operazione è $O(N^2)$. Perciò conviene effettuare tale operazione nel dominio delle frequenze calcolando la trasformata di Fourier inversa (IFFT) del cross-spectrum. In questo modo la complessità si riduce a $O(N \log N)$, a costo di una approssimazione che risulta tuttavia assolutamente accettabile. Ora la cross-correlation è espressa come

$$R_{ij}(\tau) = IFFT\left\{\sum_{k=0}^{N-1} X_i[n]\, X_j[n-\tau]^* \exp\frac{j2\pi k\tau}{N}\right\} \tag{2.2}$$

dove $X_i[n]$ è la trasformata di Fourier veloce (FFT) di $x_i[n]$ e $X_i[n]\, X_j[n-\tau]^*$ è il cross-spectrum di $x_i[n]$ e $x_j[n]$.
Come ultima operazione per trovare il TDOA, è sufficiente calcolare il massimo assoluto nella funzione di cross-correlation e fare il rapporto con la frequenza di campionamento Fs del dispositivo utilizzato, nel nostro caso 16KHz per il Kinect v1. Perciò il TDOA stimato per una coppia di microfoni i e j è:

$$R_{ij}(\tau) = \frac{\max\left(IFFT\left\{\sum_{k=0}^{N-1} X_i[n]\, X_j[n-\tau]^* \exp\frac{j2\pi k\tau}{N}\right\}\right)}{Fs} \tag{2.3}$$

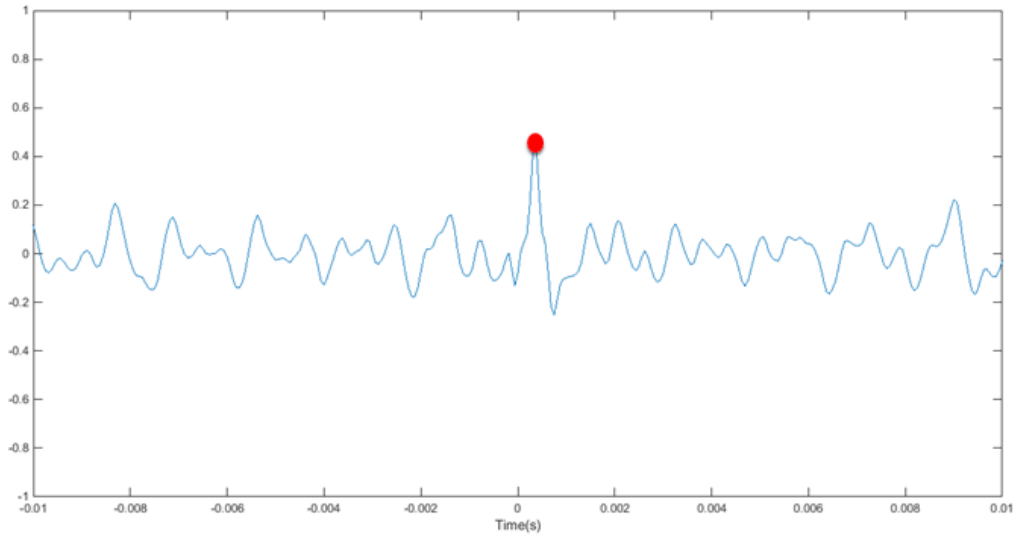


Figura 2.5: Esempio di TDOA stimato dalle registrazioni audio di una coppia di microfoni del Kinect. Il punto rosso evidenzia il picco massimo rilevato nel cross-correlation.

Nella Fig.2.5 viene mostrato un esempio del calcolo della stima di TDOA tra due registrazioni audio. L'intera operazione espressa dalla formula 2.3 è ripetuta per

ogni coppia di microfoni. Quindi se si avesse a disposizione un array di N microfoni lineari, l'intero costo computazionale del calcolo dei relativi TDOA sarebbe di $O(N * N \log N)$, essendo N-1 le coppie utilizzate per il calcolo dell'angolazione.
Per velocizzare in modo considerevole il tempo di computazione, si può limitare l'intervallo di appartenenza del TDOA τ. Infatti il caso di TDOA massimo si verifica quando l'onda sonora viaggia lungo la direzione corrispondente alla disposizione lineare dei 4 microfoni (cioè a 0° o 180°). Se così fosse, in base alla distanza d tra i microfoni di una determinata coppia e alla velocità del suono c, si calcola il TDOA massimo rilevabile da una coppia di microfoni come d/c. Applicando ora il cross-correlation su ciascuna coppia i limitando la ricerca all'intervallo [-maxTDOA(i), maxTDOA(i)], si rende più efficiente la computazione.

2.2.3 Calcolo dell'angolazione

Determinati i TDOA stimati per ciascuna coppia di microfoni, non resta che procedere al calcolo dell'angolazione della sorgente sonora rispetto al dispositivo di ricezione audio. [8] effettua tale calcolo tramite un sistema di equazioni lineari ma, date le caratteristiche del dispositivo utilizzato, verrà utilizzata una semplificazione [9]: la sorgente sonora risulta essere nel cosidetto "far-field". Si tratta di supporre che la distanza della sorgente sonora dal dispositivo di ricezione, quindi dai microfoni, sia di molto superiore all'apertura dell'array dei microfoni. In questo caso l'onda sonora risulterebbe pressoché piana nelle vicinanze dei microfoni, mentre nel caso opposto sarebbe sferica. Questa assunzione risulta essere piuttosto ipotizzabile visto che la distanza massima tra un microfono ed un altro risulta essere di 22,6 cm e ipotizziamo che la sorgente sonora non venga riprodotta al ridosso del Kinect.
Data questa ipotesi, il modello per ricavare l'angolazione dato il TDOA ΔT_{ij} di una determinata coppia di microfoni è mostrata nella Fig.2.6.

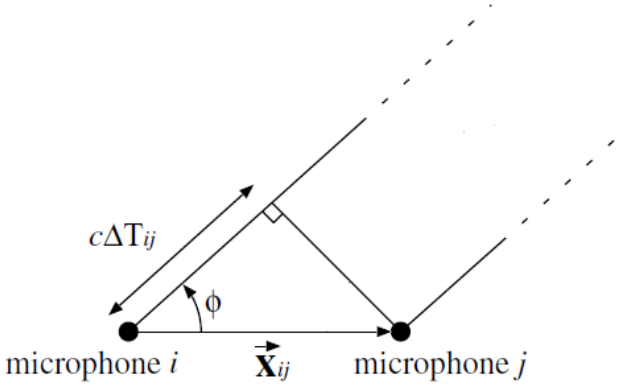


Figura 2.6: Modello semplificato per ricavare l'angolo di ricezione.

Come si può notare, si assume che l'onda arrivi "linearmente" al ricevitore cosicché si possa determinare lo spazio $c * \Delta T_{ij}$ compiuto dall'onda sonora per raggiungere il microfono i rispetto al microfono j (è possibile che ΔT_{ij} risulti negativo, quindi ovviamente nell'immagine bisogna invertire le denominazioni dei microfoni). Consi-

derando la velocità del suono c in condizione di temperatura ambiente (343,8 m/s), il calcolo dell'angolazione ϕ viene effettuato nel seguente modo:

$$\phi = \arccos\left(\frac{c * \Delta T_{ij}}{\|X_{ij}\|}\right) * \frac{180}{\pi} \tag{2.4}$$

dove $\|X_{ij}\|$ è la distanza tra i microfoni della coppia considerata. Quest'ultima formula viene applicata in ciascuna coppia per ottenere le angolazioni stimate. Per ottenere l'angolazione finale stimata, viene utilizzata la seguente formula:

$$\phi = \alpha_1\phi_1 + \alpha_2\phi_2 + \alpha_3\phi_3 \tag{2.5}$$

dove i si riferisce alla specifica coppia (1 per coppia (1,2), 2 per coppia (1,3), 3 per coppia (1,4)) e $0 \leq \alpha_i \leq 1$ è il peso relativo alla coppia che si vuole applicare. Ovviamente deve valere $\sum_{i=1}^{3} \alpha_i = 1$.

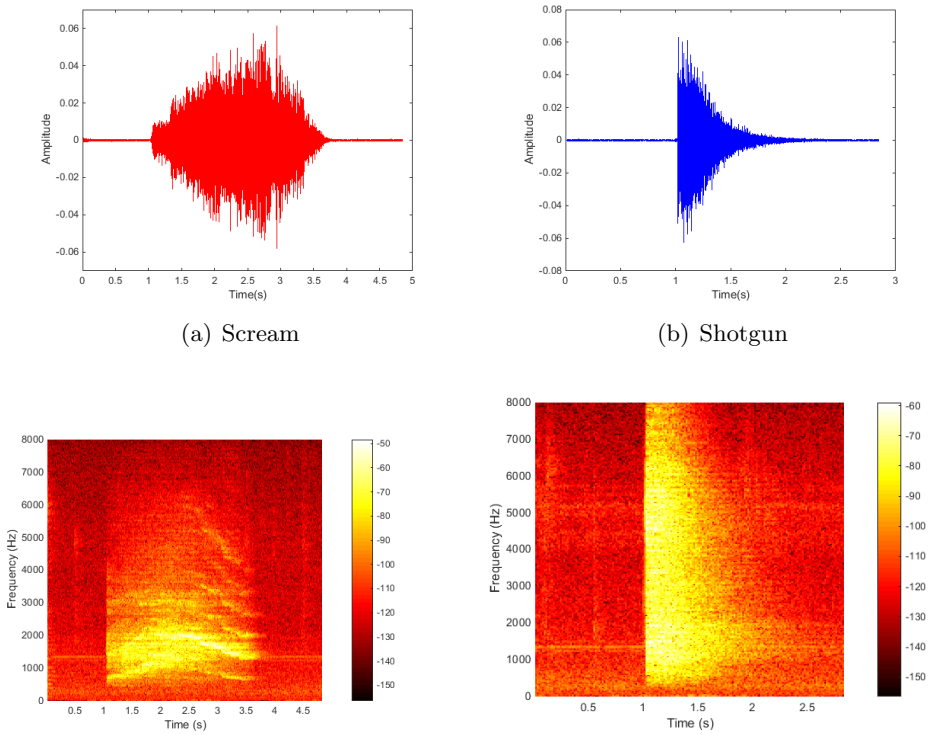
Capitolo 3

Riconoscimento di eventi sonori

Il Capitolo 2 si era concluso con l'effettiva localizzazione, intesa come angolazione rispetto al ricevitore audio, della sorgente sonora. Ora nel presente Capitolo verrà descritto il processo di riconoscimento della sorgente sonora. In particolare l'obiettivo è riuscire a distinguere se la sorgente sonora appartenga ad una categoria di nostro interesse oppure no.

3.1 Eventi sonori

Esistono molti eventi sonori che nella storia dell'arte sono stati oggetto di riconoscimento sonoro. Certamente il più famoso è il riconoscimento vocale [10] che ha numerosi campi applicativi tra cui: comandi vocali, traduzione automatica e "speech-to-text". In questa tesi verranno considerati due eventi sonori in particolare: le urla prodotte da persone ("Scream") e gli spari prodotti da armi da fuoco ("Shotgun").
Nella Fig.3.1 (a-b) sono mostrati esempi rispettivamente di scream e di shotgun nel dominio del tempo. In entrambi i casi si nota come i tempi di attacco siano molto brevi. Tuttavia se nel caso dello scream trascorre un relativo intervallo di tempo prima di raggiungere livelli medio-alti di ampiezza, nel caso dello shotgun si passa da livelli di ampiezza bassi tipici di rumore di fondo al picco massimo, corrispondente all'effettivo colpo di sparo, in pochissimo tempo (siamo nel range di centinaia di microsecondi all'incirca). Inoltre, nel caso dello shotgun, la fase di sustain è pressochè nulla e di conseguenza la fase di decay inizia non appena si conclude la fase precedente di attack.
Nella Fig.3.1 (c-d) sono invece mostrati i rispettivi spettrogrammi ottenuti nei casi dei due eventi sonori. È evidente la differenza di range di frequenze nei due casi: nello scream il range è limitato all'intervallo [1KHz, 3KHz], mentre nello shotgun il range di frequenze è pressochè illimitato, essendo un suono molto simile ad un rumore.
Scelti i tipi di eventi sonori, il riconoscimento sonoro viene attuato secondo il diagramma di flusso di Fig.3.2. Iniziando dalla traccia audio originale del suono prodotto dalla sorgente sonora, il primo passo consiste nell'estrarre le features audio, cioè le informazioni nel dominio del tempo e della frequenza che possano servire per la buona riuscita del riconoscimento sonoro (che sarà il tema della Sezione 3.2). Le features audio costituiscono l'input per l'effettivo processo di riconoscimento sonoro



(a) Scream (b) Shotgun

(c) Scream (d) Shotgun

Figura 3.1: Esempi di scream e shotgun nel dominio del tempo (a-b) e rispettivi spettrogrammi (c-d). Per questi ultimi è stato impostato l'overlap tra i frames a 256, la lunghezza di FFT a 1024, ed è stata utilizzata la finestra di Hamming.

(descritto nella Sezione 3.3), che fornirà in output un giudizio sull'appartenenza o meno del suono originale ad una delle due tipologie. Nel nostro caso, il modello di riconoscimento è in grado di distinguere tra scream e suono randomico, e tra shotgun e suono randomico.

3.2 Audio features

Il concetto di estrazione delle audio feature ha avuto origine durante le ricerche di speech processing, all'inizio per motivi di compressione dei dati e poi per speech recognition e speech synthesis. Alla fine degli anni '90 tale concetto si estese anche in ambiti non legati allo speech. La buona qualità di una feature non dipende solamente dalla sua efficacia nella classificazione, ma anche nei requisiti di computazione. Per esempio features che si basano sul FFT sono ideali per applicazioni real-time, poiché la computazione di FFT non risulta essere molto onerosa.
Le features si dividono in due gruppi:

- low-level features, che si suddividono in features nel dominio del tempo (energia media, zero-crossing rate, autocorrelazione..) e nel dominio delle frequenze (centroide spettrale, skewness, slope, MFCC...);

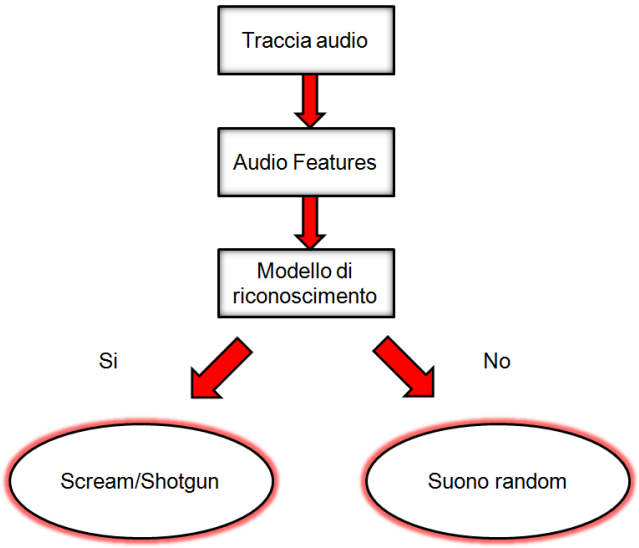


Figura 3.2: Passi nel riconoscimento sonoro.

- mid-level features, che contengono informazioni più ad alto livello (Chromagram, Keystrength, Tempo...).

In questa tesi verrà considerato solamente il primo gruppo di features, poiché gli eventi sonori esaminati non presentano alcuna caratteristica di interesse a medio-alto livello. Inoltre si sono scartate feature troppo sensitive a diversi livelli di SNR [11]. Di seguito sono elencate le feature scelte con relativa descrizione. Esse si sono basate sull'analisi, contenuta in [11], delle feature più significative per il riconoscimento degli eventi sonori considerati.

3.2.1 Zero-Crossing Rate

Zero-Crossing Rate (ZCR) rappresenta il numero di volte che il segnale audio attraversa l'asse corrispondente ad ampiezza nulla. Nella pratica corrisponde al numero totale di cambiamenti di segni del segnale tra due frame consecutivi. Nel caso di suoni non parlati tale valore è molto alto, mentre nel caso opposto di norma è basso.

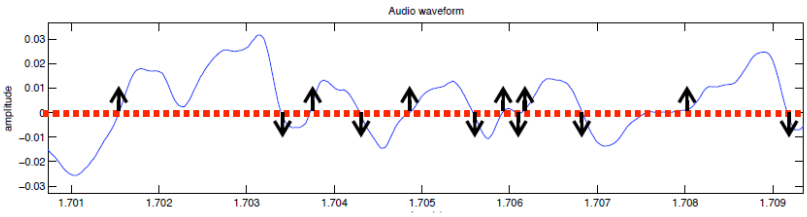


Figura 3.3: Esempio di Zero-Crossing Rate.

3.2.2 Centroide Spettrale

Il centroide spettrale (SC) è il baricentro dello spettro audio. È calcolato considerando lo spettro come una distribuzione dove i valori sono le frequenze e le probabilità sono le ampiezze normalizzate del segnale. Dato un segnale audio di N frame,

$$SC = \frac{\sum_{k=0}^{N-1} k|X[k]|}{\sum_{k=0}^{N-1} |X[k]|} \cdot \frac{Fs}{N} \tag{3.1}$$

3.2.3 Spectral Flatness Measure

Il Spectral Flatness Measure (SFM) [12] è definito come il rapporto tra la media geometrica e la media aritmetica della potenza spettrale. Data la distribuzione di potenza spettrale SPD,

$$SC = \frac{\sqrt[N]{\sum_{n=0}^{N-1} SPD(n)}}{\sum_{n=0}^{N-1} SPD(n)} \cdot \frac{1}{N} \tag{3.2}$$

Un alto valore di SFM indica che la potenza spettrale è distribuita lungo tutta la banda, tipico di un segnale audio simile al rumore bianco, mentre un basso valore indica che la potenza spettrale è concentrata in una sottobanda, tipico di un suono armonico.

3.2.4 Spectral Skewness

Il Spectral Skewness fornisce una misura dell'assimetria della distribuzione intorno al valore medio. Valori positivi indicano che pochi sample sono inferiori al valore medio e la distribuzione è concentrata nella parte sinistra, viceversa con valori negativi. Valore vicini allo zero indicano una distribuzione pressoché simmetrica.

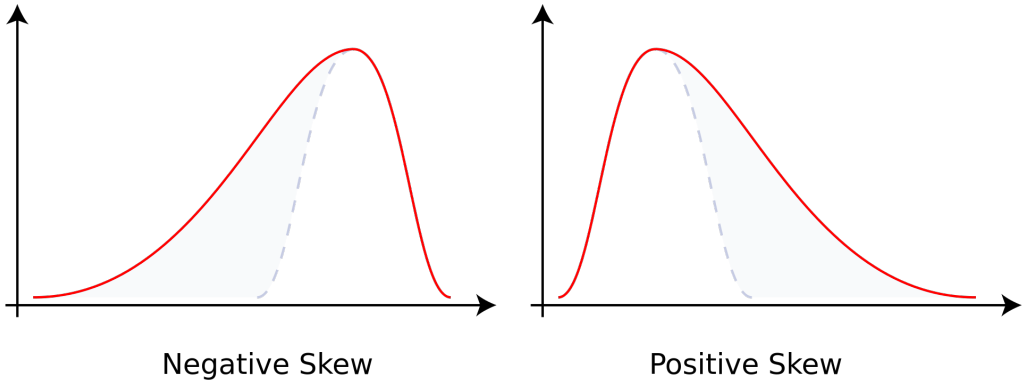


Figura 3.4: Spectral Skewness negativo e postivo.

3.2.5 Attack Slope e Decrease Slope

Lo Spectral Slope è la quantità di decrescita dell'ampiezza spettrale lungo l'asse delle frequenze. È calcolato con la regressione lineare dell'ampiezza spettrale. In particolare, considerando l'inviluppo ADSR, nel lavoro di tesi tale feature viene calcolata nella fase di attacco e nella fase di decrease.

3.2.6 MFCC

Gli MFCC sono legati alla stima dell'inviluppo spettrale del segnale audio. Considerando un segnale $y[n] = x[n] * h[n]$, spesso è utile isolare l'audio in input $x[n]$ e la risposta impulsiva $h[n]$. Tale separazione risulta evidente utilizzando il cesptrum. Tenendo conto dello spettro $Y[K]$, si applica il logaritmo, vista la proprietà di convertire somme in prodottti. In questo modo si ottiene $\log|Y[K]| = \log|X[K]| + \log|H[K]|$. In particolare, analizzando $\log|Y[K]|$, si nota un'oscillazione veloce per le componenti armoniche, legata a $X[K]$, e una più lenta dovuta alla risposta impulsiva. Applicando la DFT inversa, nell'intervallo positivo, si ottiene l'inviluppo spettrale.
Utilizzando gli studi riguardanti la psicoacustica [13], si rivela utile applicare la cosiddetta scala mel, una trasformazione non lineare nel dominio delle frequenze calcolata nel seguente modo [14]:

$$mel(f) = \begin{cases} f & se\ f \leq 1kHz \\ 2595 \log_{10}\left(1 + \frac{f}{700}\right) & se\ f > 1kHz \end{cases} \tag{3.3}$$

Nella Fig.3.5(a) viene mostrata una rappresentazione grafica della funzione $mel(f)$. Si nota come nelle frequenze più basse il comportamento risulti lineare, mentre nelle frequenze superiori il comportamento sia logaritmico. In tal modo si tenta di simulare la ricezione umana del suono, che risulta dipendente dalla frequenza del suono ricevuto.

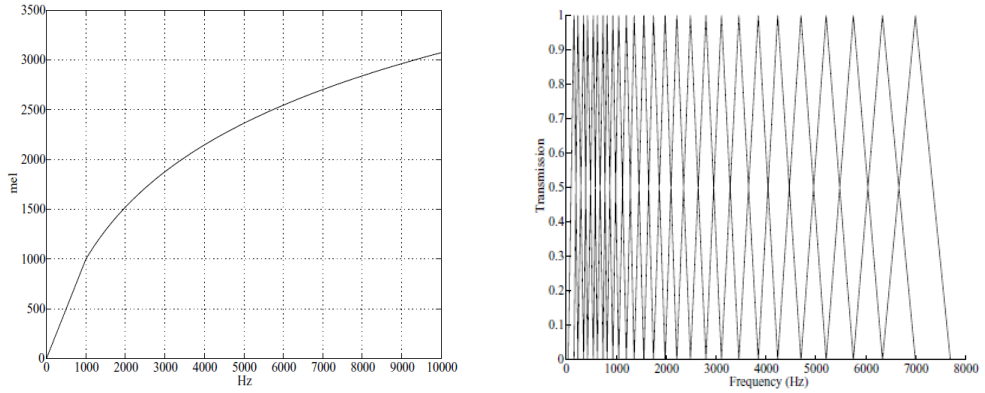


Figura 3.5: Rappresentazione grafica della funzione $mel(f)$ (a) e i filterbank nella scala mel (b) [14].

Per applicare la scala mel al cepstrum sono utilizzati filterbank triangolari banda passante (vedesi Fig.3.5(b)), con frequenze centrali in K valori mel. In particolare

ogni filtro è caratterizzato da una banda il cui valore è il doppio rispetto al filtro precedente (il primo filtro parte da 0). Utilizzando il mel-cesptrum, si è in grado di stimare l'inviluppo spettrale dell'output del filterbank.
Se Y_n è il logaritmo dell'energia uscente dal canale n, tramite la trasformata discreta del coseno DCT, si calcolano i coefficienti mel-cepstrali MFCC in questo modo:

$$c_k = \sum_{n=1}^{N} Y_n \cos\left[k\left(n - \frac{1}{2}\frac{\pi}{N}\right)\right] \qquad k = 0, \ldots, K \tag{3.4}$$

Tramite i MFCC è possibile stabilire l'inviluppo spettrale del segnale in input.

3.3 Machine Learning

Dopo aver selezionato le feature di interesse per la classificazione, è necessario stabilire quale modello di riconoscimento applicare per l'analisi delle feature e la classificazione dei file audio in input. Tutto questo rientra nell'ambito del Machine Learning, una branca della scienza che si occupa dello studio di modelli che automaticamente forniscono in output informazioni utili riguardo i dati in input. In particolare per lo scopo della tesi si analizzeranno due sottobranche del Machine Learning: classificazione supervisionata e classificazione non supervisionata.
Nella prima categoria, il dataset di training contiene campioni con label (etichette) già assegnate che determinano l'appartenenza di tale campione ad una determinata categoria. Lo scopo è costruire un modello di predizione in base al training set, e di assegnare label ai campioni di un test fornito in input.
Nella seconda categoria, il procedimento è il medesimo, ma il dataset è sprovvisto di etichette. Perciò il modello, durante la fase di training, cerca di assegnare le etichette ai dati in input in base alle somiglianze/differenze presenti nel dataset. Spesso ci si riferisce a questa categoria come clustering.
Nel lavoro di tesi sono stati utilizzati modelli provenienti dalla classificazione supervisionata, più una versione supervisonata di clustering [15], dove il modello cerca di raggruppare in ciascun cluster ogni campione del dataset di una determinata label. Infatti il dataset raccolto per il lavoro di tesi è già suddiviso nelle varie label di interesse, quindi non si considera un dataset sprovvisto di label.
Nelle seguenti sottosezioni verrano elencati e descritti i principali modelli di Machine Learning utilizzati.

3.3.1 C4.5

C4.5 [16] è un algoritmo sviluppato da Ross Quinlan e si tratta dell'evoluzione dell'algoritmo ID3, sviluppato dal medesimo. Appartiene alla famiglia degli alberi di decisione dove ogni nodo interno, inclusa la radice, rappresenta una condizione sugli attributi per separare i vari record e ad ogni foglia viene associata una determinata label. Dato un dataset S, C4.5, come gli altri algoritmi basati sugli alberi, costruisce l'albero tramite la seguente strategia divide-and-conquer [17]:

- se tutti i campioni di S appartengono ad una medesima label o il loro numero è sufficientemente basso, all'albero viene aggiunta una foglia e associata la label con la frequenza maggiore all'interno di S;

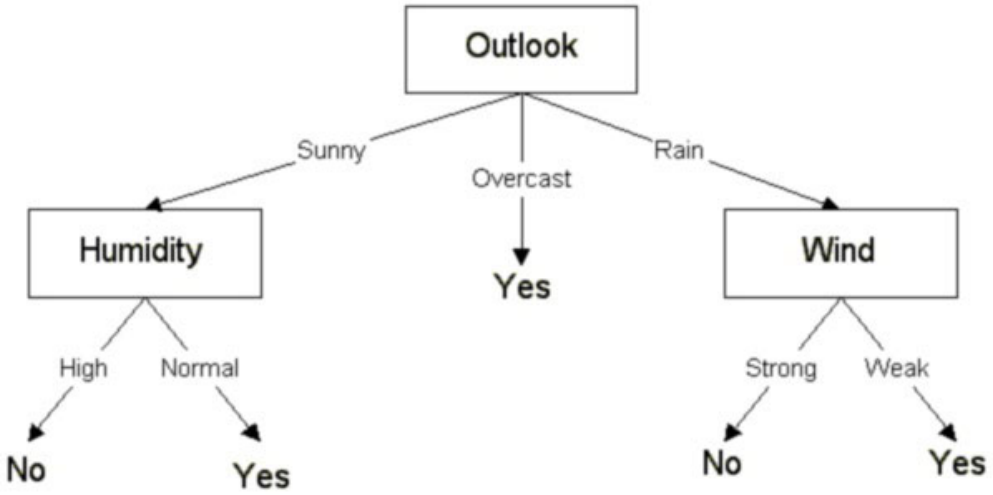


Figura 3.6: Esempio di albero di decisione.

- altrimenti si sceglie un test su un singolo attributo, si costruisce la radice basata su questo test, e si partiziona l'albero in diversi sottoalberi in base ai vari output che il test produce. Ad ogni sottoalbero è associata una partizione del dataset S e viene eseguita ricorsivamente la medesima procedura.

C4.5 utilizza due rank per scegliere il migliore test da eseguire in un determinato nodo interno. Il primo rank è l'information gain Δ_{info}, descritto dalla seguente formula:

$$\Delta_{info} = I(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{3.5}$$

dove $I(\cdot)$ è la misura di impurità di un determinato nodo, N è il numero totale di campioni nel nodo genitore, K è la dimensione dell'attributo, e $N(v_j)$ è il numero di record associati al nodo figlio v_j. Il test che massimizza Δ_{info} avrà il rank più alto. L'information gain ha il difetto di tendere a premiare attributi che producano un alto numero di sottoalberi (split) contenenti un numero non sufficientemente alto di record per determinare una buona predizione.
Il secondo rank ha il compito di risolvere tale difetto ed è il gain ratio. Esso semplicemente divide l'information gain per il Split Info, dove quest'ultimo risulta essere uguale a $-\sum_{i=1}^{s} P(v_i) \log_2 P(v_i)$, con s il numero totale degli split. Tale rank determina la bontà dello split, poiché un numero alto di split prodotto da un attributo diminuisce il gain ratio.

3.3.2 Random Forest

L'algoritmo fu sviluppato da Leo Breiman [18] e Adele Cutler [19] e rappresenta un notevole miglioramento rispetto agli alberi di decisione base. Infatti uno dei principali problemi di questi classificatori consiste nell'overfitting: l'albero di decisione,

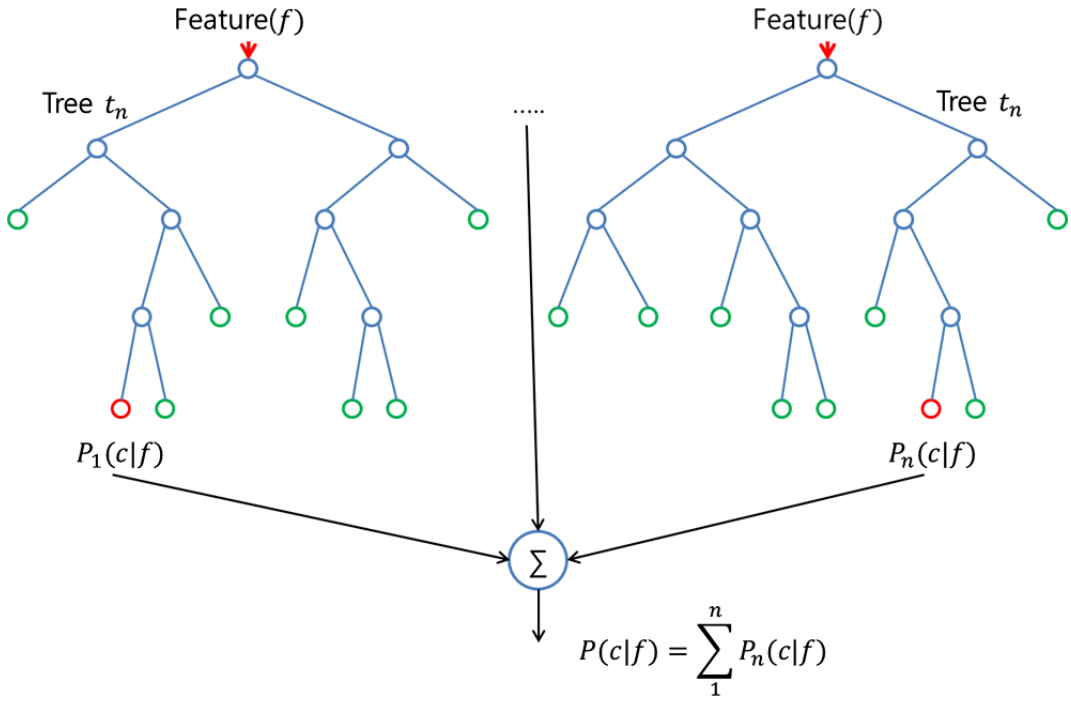


Figura 3.7: Random Forest.

se nel training set è presente un rumore tra i dati o mancano alcuni dati importanti, risente molto in termini di precisione. A questo e ad altri problemi l'algoritmo Random Forest cerca di rimediare. Esso si basa ancora sull'algoritmo dell'albero di decisione per la classificazione, tuttavia Random Forest costruisce più alberi (il cui numero è scelto dall'utente), dove ciascuno analizza un input fornito e produce come risultato una classificazione. Alla fine, facendo la media delle varie predizioni o scegliendo la predizione più comune tra i vari alberi, si sceglie la predizione più opportuna.
Per poter creare gli alberi di decisione si utilizza la tecnica di Bootstrap: dall'intero training set si seleziona, per ogni albero che si vuole creare, un suo sottoinsieme, dove i dati possono essere duplicati ("Sampling with replacement"). Poi per ogni nodo si sceglie in modo casuale dall'insieme totale degli attributi un numero m di questi, specificato all'inizio dall'utente, e il migliore split (stabilito tramite la misura Gini che Random Forest utilizza) è usato per proseguire nella creazione dell'albero. Si noti che nel Random Forest non viene applicato alcun metodo di pruning (a meno che non sia specificato dall'utente) e quindi ogni albero viene costruito con la massima profondità possibile. In totale, circa un terzo dei dati presenti nel training set non viene selezionato per ogni albero. Alla fine ogni record del training set viene classificato sugli alberi che non si sono basati su di esso per la propria costruzione (out-of-bag examples) e, confrontando il responso di tali alberi con la vera classe, si calcola l'oob error rate sulla totalità dei record. Grazie a questo aspetto, non serve l'utilizzo di metodi di valutazione come per esempio il Cross-Validation, poiché un metodo di valutazione è già presente nel Random Forest.
L'oob viene utilizzato anche per trovare l'importanza di un attributo. Infatti basta

permutare il valore dell'attributo nei vari oob e si calcola l'oob error rate. Poi si calcola per ogni albero la differenza di errore tra prima e dopo la permutazione e si opera la media su tutti gli alberi. Tale dato mostra quanto rilevante sia tale attributo.
Un ulteriore aspetto presente nel Random Forest è l'utilizzo dei proximity (numero di volte in cui due attributi si trovano nello stesso nodo terminale) degli output. Essi possono essere usati per creare sample da porre nel training set in modo da riempire alcuni casi non presenti, per localizzare outlier e creare grafici di topologia in dimensione bassa.
Studi hanno mostrato come gli errori compiuti dal Random Forest dipendano da due fattori:

- la correlazione tra ciascuna coppia di alberi creati nella foresta (più la correlazione aumenta, più aumenta l'errore).
- la "robustezza" di ogni albero di decisione creato (alberi caratterizzati da un error rate basso diminuiscono l'error rate totale del Random Forest).

Di grande importanza risulta la scelta del numero di attributi da selezionare ad ogni nodo. Infatti riducendo tale numero, si riduce la correlazione ma anche l'efficienza dei singoli alberi (aumentandolo si hanno gli effetti opposti). Perciò, controllando anche l'oob error rate, bisogna cercare il valore ottimale che sia di compromesso tra i due fenomeni indicati.

3.3.3 AdaBoostM1

Questo algoritmo [20] fa parte di una particolare categoria di metodi di classificazione, gli "Ensemble methods". Tali metodi costruiscono un set di classificatori base dai dati di training e assegnano la label in base ai voti sulle predizioni compiute da ciascun classificatore. AdaBoostM1 si basa sulla tecnica del boosting, che cambia la distribuzione degli esempi di training così che i classificatori base si concentrino su esempi difficili da classificare. In particolare AdaBoostM1 [21] è una delle versioni dell'algoritmo base AdaBoost.
Come descritto nell'Alg.2, in una serie di round AdaBoostM1 chiama un classificatore base ("WeakLearn"), dove nel round t il booster fornisce al WeakLearn una distribuzione D_t rispetto al training set. In particolare la distribuzione iniziale D_1 è uniforme lungo il training set (quindi $D_1(i) = 1/m$ per ciascuna istanza i). WeakLearn risponde fornendo una ipotesi h_t di label su tutto il training set. Se l'errore compiuto, rispetto alla distribuzione D_t, risulta essere minore di 0.5, allora si aggiorna $D_t \rightarrow D_{t+1}$ e si richiama WeakLearn, altrimenti si reinizializza tutto. Alla fine l'ipotesi finale h_{fin}, che fornisce in output una label, sarà l'ipotesi che massimizza la somma dei pesi delle ipotesi h_t che predicono tale label. Il threshold di errore a 0.5 è giustificato dal seguente teorema [21]: se un ipotesi h_t risulta superiore a 0.5, l'errore su h_{fin} tende velocemente a zero, portando ad una veloce convergenza dell'algoritmo. Di seguito viene esposto il teorema completo.

Teorema 1. *Supponendo che il classificatore WeakLearn generi, in una serie di T round, ipotesi con rispettivi errori ϵ_t $(t = 1, \cdots, T)$ e $\gamma_t = 1/2 - \epsilon_t$, se $\forall \epsilon_t \leq 0.5$ allora vale il seguente upper bound:*

$$\frac{1}{m} \left| \{ i : h_{fin}(x_i) \neq y_i \} \right| \leq \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leq exp\left(-2 \prod_{t=1}^{T} \gamma_t^2 \right).$$

Algorithm 2 Algoritmo AdaBoostM1 [21].

1: $\{(x_1, y_2), \cdots, (x_m, y_m)\}$ sequenza di m campioni con label $y_i \in Y = \{1, \cdots, k\}$;
2: **WeakLearn** modello base di classificazione;
3: T numero di iterazioni;
4: D distribuzione rispetto al training set;
5: **procedure** MAIN
6: Inizializza $D_1(i) = 1/m$ per $\forall i$;
7: **for** $t = 1, 2, \cdots, T$ **do**
8: Chiama **WeakLearn** su D_t che ritorna l'ipotesi $h_t : X \to Y$;
9: Calcola l'errore di h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$;
10: **if** $\epsilon_t > 1/2$ **then**
11: $T = t - 1$;
12: Esci dal loop;
13: **else**
14: $\beta_t = \epsilon_t / (1 - \epsilon_t)$;
15: Aggiorna $D_t \to D_{t+1} = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & se\ (h_t(x_i) = Y_i) \\ q & altrimenti \end{cases}$
dove Z_t è una costante di normalizzazione scelta in modo che D_{t+1} sia una distribuzione;
16: Ritorna l'ipotesi finale $h_{t+1}(x) = arg \max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$;

3.3.4 AODE

Averaged one-dependence estimator (AODE) [22] si fonda sul noto classificatore Naive Bayes. Come suggerisce il nome di quest'ultimo, si basa sul teorema sulle probabilità di Bayes. Se si considera $\mathbf{X}$ il set di attributi e $\mathbf{Y}$ l'insieme delle label, allora sfruttando il teorema si ottiene:

$$P(\mathbf{Y}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{Y}) P(\mathbf{Y})}{P(\mathbf{X})} \tag{3.6}$$

Quindi, essendo $P(\mathbf{X})$ costante, $P(\mathbf{Y}|\mathbf{X})$ è direttamente proporzionale a $P(\mathbf{X}|\mathbf{Y}) P(\mathbf{Y})$. In Naive Bayes si assume l'indipendenza condizionata tra gli attributi, perciò

$$P(\mathbf{X}|\mathbf{Y}) = \prod_{i=1}^{n} P(X_i|\mathbf{Y}) \tag{3.7}$$

Assumendo tale ipotesi è sufficiente calcolare $P(X_i|\mathbf{Y})$ per ogni i e $P(\mathbf{Y})$ per poter determinare $P(\mathbf{Y}|\mathbf{X})$. La label y più probabile per l'istanza X sarà quella che massimizza $P(y|X)$.
Uno dei difetti del Naive Bayes consiste nell'assunzione dell'indipendenza tra gli attributi, che spesso non sussiste. AODE è stato sviluppato con lo scopo di ovviare a tale difetto diminuendo il costo computazionale di metodi simili come LBR e SP-TAN. Dati gli attributi $x_1, \cdots, x_n$ ed una label y per ciascuno $1 \leq i \leq n$, vale

$$P(y, x_1, \cdots, x_n) = P(y, x_i) P(x_1, \cdots, x_n|y, x_i) \tag{3.8}$$

AODE considera $x_1, \cdots, x_n$ indipendenti rispetto a y e a x_i, perciò ne consegue che

$$P(y, x_1, \cdots, x_n) = P(y, x_i) \prod_{j=1}^{n} P(x_j|y, x_i) \tag{3.9}$$

Tale concetto è il One Dependence Estimator (ODE), una ipotesi di indipendenza più debole rispetto all'assunzione posta dal classificatore Naive Bayes. In base all'ODE si ottiene la seguente formula:

$$P_e(y, x_1, \cdots, x_n) = \frac{\sum_{i:1 \leq i \leq n \wedge F(x_i) \geq m} P_e(y, x_i) \prod_{j=1}^{n} P_e(x_j|y, x_i)}{\sum_{y' \in Y} \sum_{i:1 \leq i \leq n \wedge F(x_i) \geq m} P_e(y, x_i) \prod_{j=1}^{n} P_e(x_j|y, x_i)} \tag{3.10}$$

dove $P_e(\cdot)$ è la stima di $P(\cdot)$, $F(\cdot)$ è la frequenza dell'attributo nel dataset e m è una frequenza minima, specificata dall'utente, del valore dell'attributo affinché venga accettato. Il denominatore è giustificato per una questione di normalizzazione su tutte le label.

3.3.5 GMM-EM

I Gaussian Mixture Model (GMM) [23] appartengono alla famiglia delle misture di distribuzioni, che sono combinazioni lineari di distribuzioni semplici, e sono applicate nel campo del clustering. Nel caso di GMM la distribuzione utilizzata è la gaussiana. In particolare una mistura di K gaussiane è definita nel seguente modo:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \sigma_k) \tag{3.11}$$

con μ_k vettore media, σ_k matrice di covarianza e $N(\cdot)$ una distribuzione gaussiana. I π_k sono i cosidetti mixing coefficient applicati a ciascuna distribuzione gaussiana. Nel contesto del Machine Learning, il vettore x corrisponde al vettore degli attributi e quindi $p(\mathbf{x})$ equivale alla probabilità che un'istanza possieda determinati valori negli attributi secondo la distribuzione gaussiana.

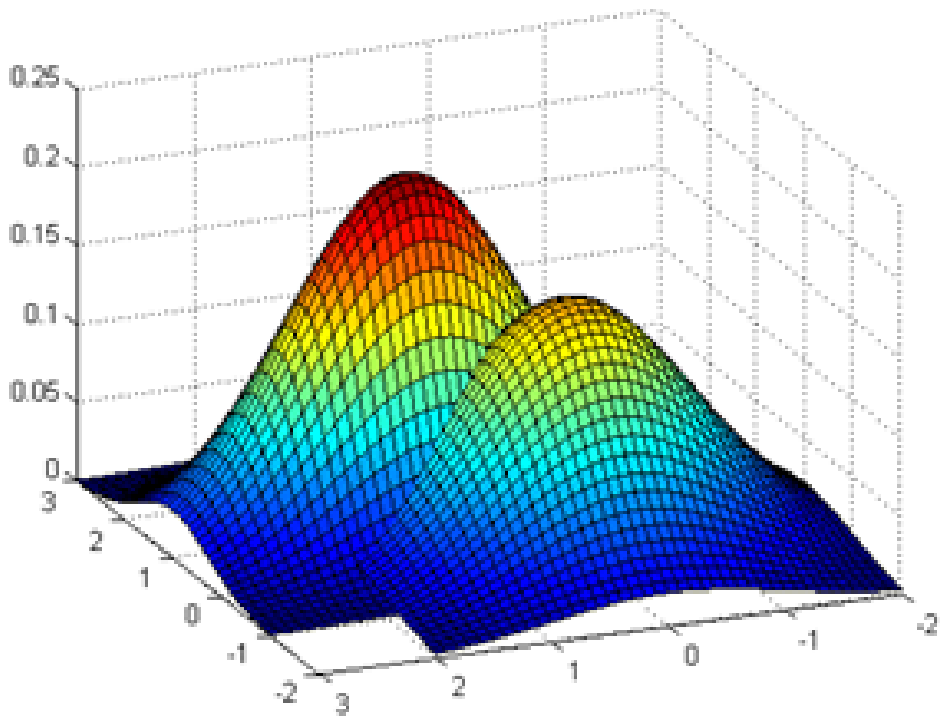


Figura 3.8: Gaussian Mixture Model.

Se si indica con $\mathbf{z}$ la variabile della label, la probabilità a posteriori $p(\mathbf{z}|\mathbf{x})$ è definita, tramite l'utilizzo del teorema di Bayes, come:

$$p(z_k = 1|\mathbf{x}) = \frac{p(\mathbf{x}|z_k = 1)\, p(z_k = 1)}{p(\mathbf{x})} = \frac{\pi_k N(\mathbf{x}|\mu_k, \sigma_k)}{\sum_{j=1}^{K} \pi_j N(\mathbf{x}|\mu_j, \sigma_j)} \tag{3.12}$$

in cui π_k sono le probabilità che $z_k = 1$
Tramite l'utilizzo dei GMM l'obiettivo è associare i dati del training set nelle rispettive componenti gaussiane, cioè i cluster, in modo da massimizzare le probabilità a posteriori. Perciò è necessario definire tre parametri per ogni componente gaussiana/cluster k-esima:

- il vettore media μ_k;
- la matrice di covarianza σ_k;
- la probabilità a priori π_k, cioè il peso assegnato ad una determinata label.

Per tale contesto, viene utilizzato l'algoritmo EM [24], un metodo iterativo per trovare il maximum likelihood o il massimo a posteriori dei parametri di modelli statistici. EM è diviso in due parti, lo step E e lo step M.

- Nell' *Expectation step* (E) si utilizzano i valori attuali di μ_k, σ_k, π_k per calcolare le probabilità a posteriori, che vengono utilizzate per assegnare i campioni del dataset nelle componenti gaussiane che massimizzano la rispettiva probabilità.
- Nel *Maximization step* (M) si utilizzano le probabilità a posteriori e le nuove assegnazioni dei campioni per ridefinire i parametri delle componenti gaussiane.

Alla fine di ogni iterazione dell'algoritmo EM la log-likelihood dei parametri aumenta. I fattori di terminazione di EM sono il limite del numero di iterazioni da compiere e la soglia di miglioramento della log-likelihood.

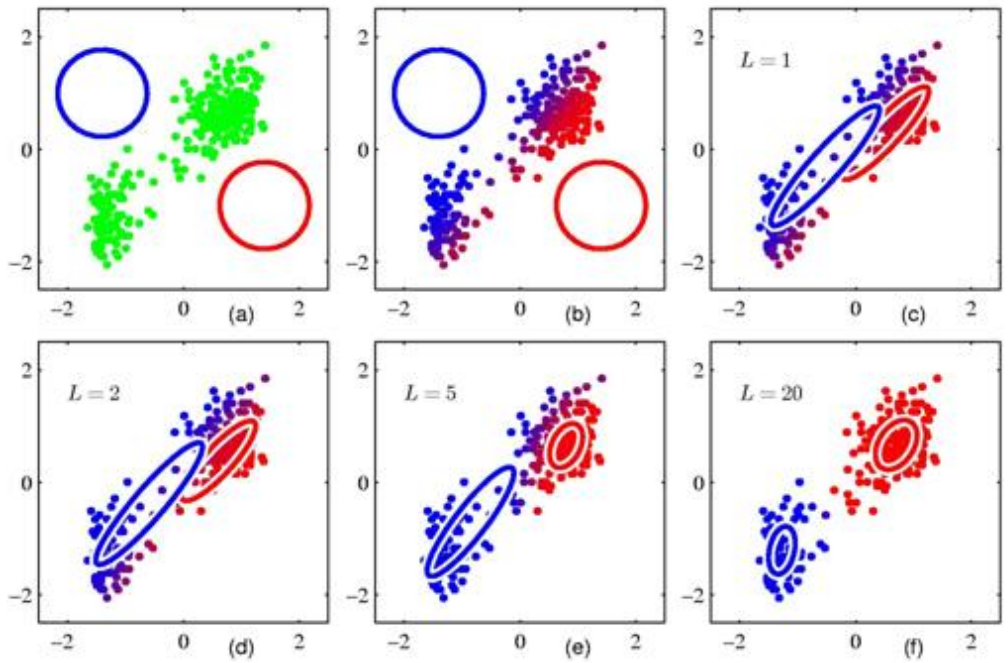


Figura 3.9: Esempio di applicazione dell'algoritmo EM.

Come accennato all'inizio di questa Sezione, i campioni del dataset nel lavoro di tesi sono già provvisti della rispettiva label. Perciò si è parlato di clustering supervisionato. Nel caso in questione, se consideriamo una partizione del dataset in modo che ad ogni partizione siano associati tutti i campioni di una determinata label, verrà costruito un GMM per ogni partizione. Nella valutazione del modello, per ogni GMM verrà calcolata la probabilità a posteriori sul test e la GMM con la log-likelihood più alta determinerà la corrispondente label da assegnare.

3.3.6 Metodo di validazione del classificatore

Non avendo a disposizione campioni di test al di fuori del dataset raccolto, è necessario basarsi su metodi di validazione del classificatore prescelto. Tra questi il più famoso è il k-cross-validation. Il metodo divide il training set in k partizioni di uguale dimensione e ad ogni round sceglie una partizione per il test set e le rimanenti partizioni vengono assegnate al nuovo training set. Questo procedimento è ripetuto k volte così che ogni partizione venga utilizzata esattamente una volta per il test.

3.4 Implementazione del riconoscimento

In questa Sezione verranno descritti i vari passi degli algoritmi utilizzati per attuare il riconoscimento degli eventi sonori prescelti, che si ricorda sono i scream e i shotgun. La fase di estrazione delle features audio e dell'applicazione dei GMM è stata effettuata tramite l'ambiente di lavoro Matlab, mentre l'applicazione dei rimanenti modelli di Machine Learning è stata svolta utilizzando Weka, un software open-source per l'apprendimento automatico sviluppato nell'Università di Waikato in Nuova Zelanda.

1. A partire dai file audio (suddivisi in Random Sounds, Scream, Shotgun) in base a quale dei due eventi si vuole riconoscere, si considera un nuovo insieme Random Sounds, che corrisponde all'unione del Random Sounds originale e dell'evento sonoro non considerato. Tale operazione è giustificata dalla possibilità di disporre di tutti i campioni per la fase di training e di test del modello di machine learning.

2. Per ciascuno dei file audio vengono calcolate le feature audio descritte nella Sezione 3.2. Tale fase avviene tramite la libreria MIRToolbox [25], che dispone di funzioni per l'estrazione di numerose features dai file audio (devono essere in formato WAV). In tutti i casi si sono considerati i valori medi. Di seguito sono elencate le funzioni utilizzate:

   - zero-cross rate tramite *mirzerocross*;
   - centroide spettrale tramite *mircentroid*. Per evitare la comparsa di valori NaN, si è imposto il valore MinRMS a 0 e il MaxEntropy a 1;
   - i coefficienti MFCC tramite *mirmfcc*. Si è scelto di usare i primi 13 coefficenti, valore di default della funzione;
   - spectral flatness tramite *mirflatness*;
   - spectral skewness tramite *mirskewness*;
   - attack slope tramite *mirattackslope*;
   - decrease slope tramite *mirdecreaseslope*.

3. Dopo aver raccolto le feature di tutti i file audio, si effettua una conversione di Z-score in modo che tutte le feature possano essere confrontate. Le feature convertite, che compongono l'intero dataset, vengono salvate in file di formato csv, premunendosi di inserire una riga iniziale per la definizione degli attributi e una colonna finale per l'inserimento della label.

A questo punto, in base al modello di riconoscimento scelto, sono state adoperate due diverse implementazioni. Il prossimo elenco numerato riguarda l'implementazione dei GMM.

1. Se si vuole applicare i GMM in ambiente Matlab, dopo aver caricato il dataset (si ricorda che è distinto in base al tipo di evento sonoro da riconoscere) si è implementata la k-cross validation per la validazione, con $k = 10$. Per ogni round si sceglie una partizione per il test set e le rimanenti partizioni per il training set.

2. Si divide il training set in due: training_target e training_random. Il primo include tutti i campioni riguardanti l'evento sonoro da riconoscere, il secondo include i campioni non appartenti alla prima categoria.

3. Tramite la funzione matlab *fitgmdist* si creano le rispettive GMM. Tale funzione richiede in input lo specifico training set e il numero delle componenti da utilizzare. Tale numero può essere calcolato tramite analisi AIC, PCA o sperimentazione. Avendo scelto la terza opzione, si è scelto 1 come numero delle componenti.

4. Per valutare il test set si utilizza la funzione *posterior*, che riceve in input la specifica GMM, target o random, e il test. Tra gli ouptut è presente il log-likelihood. Confrontando i due output, il valore più alto di log-likelihood determina la scelta del target da assegnare.

5. L'output finale è la Confusion Matrix. Essa è una tabella, applicabile per classificazioni binarie, che descrive il numero di istanze del test predette correttamente ed erroneamente. In particolare sono presenti quattro valori:

   - true positive (TP), il numero di istanze positive (cioè appartenenti al target) correttamente predette;
   - false negative (FN), il numero di istanze negative (cioè appartenenti ai random) non correttamente predette;
   - false positive (FP), il numero di istanze positive non correttamente predette;
   - true negative (TN), il numero di istanze negative correttamente predette.

6. Dato che il metodo di cross-validation implementato ad ogni esecuzione del programma produce partizioni diverse, la Confusion Matrix fornita in output può variare leggermente. Per stabilizzare il risultato l'intero procedimento, a partire dalla partizione del dataset, viene ripetuto per un numero definito di iterazioni. Si è deciso come numero di iterazioni 1000.

Nel caso dei rimanenti modelli si sono utilizzate le rispettive versioni implementate nel software Weka (si rammenta che in tale contesto il modello C4.5 corrisponde a J4.8 in Weka) utilizzando la 10-cross validation. Per poter utilizzare gli algoritmi è stato reso nominale l'attributo corrispondente alla label, mentre tutti gli attributi rimanenti sono stati lasciati nominali.

Capitolo 4

Risultati

Nel presente Capitolo verranno mostrati i risultati ottenuti da vari test riguardanti i due obiettivi principali prefissati. In particolare la Sezione 4.1 riguarderà la localizzazione sonora, intesa come angolazione della sorgente sonora rispetto al ricevitore audio, tramite l'array di microfoni. La Sezione 4.2 riguarderà il riconoscimento degli eventi sonori prefissati variando i modelli di predizione utilizzati.

4.1 Localizzazione sonora

4.1.1 Setup dell'esperimento

Per simulare la riproduzione degli eventi sonori a diversa angolazione rispetto al dispositivo di ricezione utilizzato (Kinect v1), è stato allestito il seguente ambiente di lavoro (si veda la Fig.4.1): alla distanza di 1m rispetto alla posizione della Kinect sono stati riprodotti i diversi eventi sonori selezionati a diversa angolazione nel range [0°, 180°]. Considerando tale distanza e l'apertura massima dei microfoni (circa 22,6 cm) l'ipotesi del far-field introdotta nel Capitolo 2 risulta essere plausibile. La riproduzione audio è stata effettuata in una stanza assente da rumore e di dimensioni tali che l'eco fosse praticamente assente. La scelta dell'assenza di rumore è giustificata, oltre che dal poter acquisire audio per i seguenti test nelle migliori condizioni possibili, dal successivo test di localizzazione sonora con aggiunta di rumore bianco presente nella Sezione 4.1.4.
Si è deciso di utilizzare come step di angolo 5° per ottenere test più precisi e verificare in quali condizioni l'algoritmo presenta prestazioni migliori e in quali prestazioni peggiori. I suoni da localizzare sono stati l'urlo di spavento di una donna e un colpo di arma da fuoco, entrambi della durata di circa 5 secondi, riprodotti dagli altoparlanti di un cellulare disposto all'angolazione richiesta. Quindi in tutto sono state effettuate 74 registrazioni di eventi sonori.

4.1.2 Esecuzione dei test su angolazione

Come descritto nel Capitolo 2, la fase di registrazione audio è stata effettuata tramite l'utilizzo di un modello di Kinect v1 collegato ad un PC con sistema operativo Linux e libreria freenect installata. Come output finale si sono ottenuti i file audio dei 4 microfoni per ciascuna delle 74 registrazioni compiute a diversa angolazione.

Successivamente per ogni quadrupla di registrazioni è stato applicato l'algoritmo di localizzazione dell'angolazione, sviluppato in ambiente Matlab.

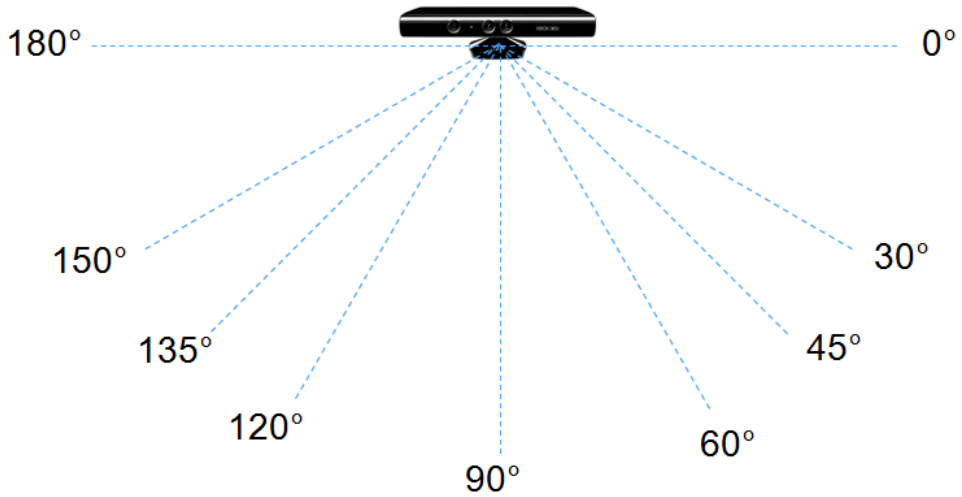


Figura 4.1: Setup dei test sulla ricezione sonora e sulla localizzazione.

4.1.3 Quantizzazione

Prima di procedere ad elencare i risultati ottenuti, è necessario presentare un problema evidenziato nei primi test sulla localizzazione. Infatti analizzando le direzioni ricavate da ciascuna coppia di microfoni si notava come, considerando un range d'angolo sufficientemente piccolo, queste spesso non si modificassero al variare dell'angolo in tale range, ma risultassero fisse. Solamente considerando range più elevati si iniziava a notare degli output di direzione differenti. Tale fenomeno è dovuto a due cause:

- la frequenza di campionamento del dispositivo (16KHz), che causa la quantizzazione del segnale audio analogico in un segnale discreto non sufficientemente dettagliato;
- l'apertura dell'array di microfoni (al massimo di 22,6 cm), che non consente di avere a disposizione un'alta precisione nella localizzazione della sorgente sonora.

A giustificazione di queste cause, la Tabella 4.1 mostra i possibili angoli quantizzati dalla coppia di microfoni (1,4) (la coppia con i microfoni più distanti tra di loro), (1,3) e (1,2) (la coppia con i microfoni più vicini). Analizzando per esempio la coppia (1,4) si nota come nei range di angoli vicini a 0° e a 180° questi risultino essere molto quantizzati, con uno step d'angolo anche di circa 15°. Considerando invece i rimanenti casi, la quantizzazione è presente ma con minore intensità (circa 6°). Tale quantizzazione viene enfatizzata considerando le altre coppie di microfoni, dove le distanze tra i microfoni risultano essere minori, comportando step di angolo più elevati.

Tuttavia è necessario menzionare il fatto che il dispositivo di ricezione audio utilizzato è stato sviluppato concentrandosi sulla componente video, e con minore importanza sulla componente audio. Inoltre, dato che per gli ambiti applicativi di videosorveglianza spesso è sufficiente individuare l'origine della sorgente sonora in un range sufficiente ampio da permettere alla camera di inquadrarla, tale quantizzazione non costituisce un grosso limite.

Angolazione stimata da (1,4)	Angolazione stimata da (1,3)	Angolazione stimata da (1,2)
0,00°	0,00°	0,00°
18,05°	24,56°	30,09°
31,16°	37,27°	43,86°
40,48°	46,99°	54,77°
48,27°	55,36°	64,37°
55,21°	62,95°	73,24°
61,62°	70,06°	81,71°
67,65°	76,86°	90,00°
73,43°	83,47°	98,29°
79,04°	90,00°	106,76°
84,54°	96,53°	115,63°
90,00°	103,14°	125,23°
95,46°	109,94°	136,14°
100,96°	117,05°	149,91°
106,57°	133,01°	180,00°
112,35°	142,73°	
118,38°	155,44°	
124,78°	180,00°	
131,72°		
139,52°		
73,43°		
148,84°		
161,95°		
180,00°		

Tabella 4.1: Elenco dell'insieme degli angoli quantizzati per le coppie di microfoni considerate

Dato che l'effetto di quantizzazione dipende dalla distanza tra i microfoni della coppia considerata, è stato deciso che per quanto riguarda i pesi delle direzione stimate sia imposto che $a_1 \leq a_2 \leq a_3$.

4.1.4 Risultati su angolazioni stimate

Nella Fig.4.2 viene mostrato un esempio di errore compiuto dal dispositivo Kinect nel rilevare le angolazioni reali. Come si poteva supporre dalle considerazioni della precedente sottosezione, nei range di angoli vicino a 0° e a 180° gli errori compiuti possono avvicinarsi a 8-10° e variano molto a distanza di poche misurazioni. Nei

restanti casi il sistema compie invece errori vicino ai 2°- 4°, risultando quindi piuttosto preciso, e tende a fornire stime più elevate rispetto al vero nel caso di angoli inferiori a 90° e il viceversa con angoli superiori a 90°.

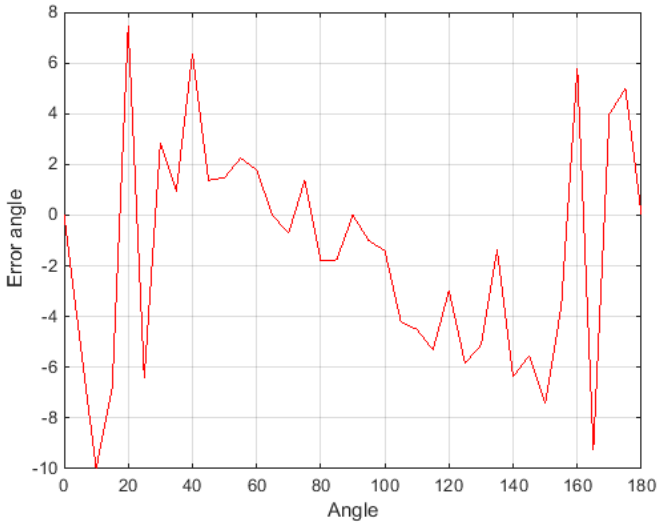
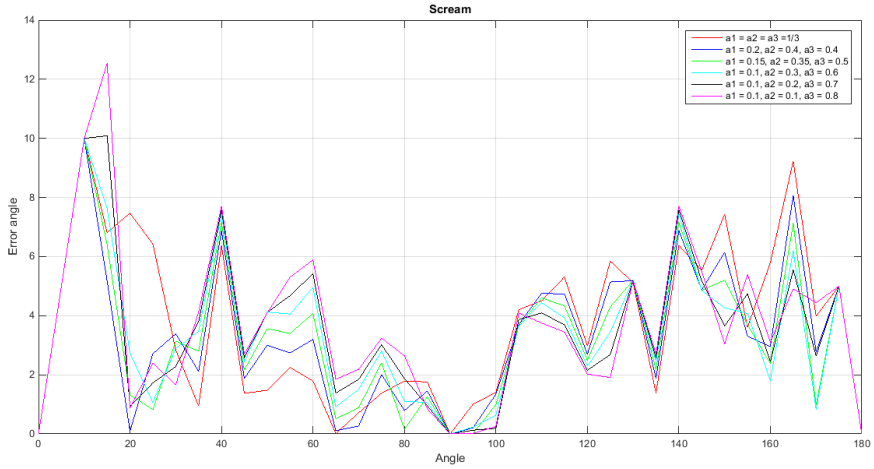
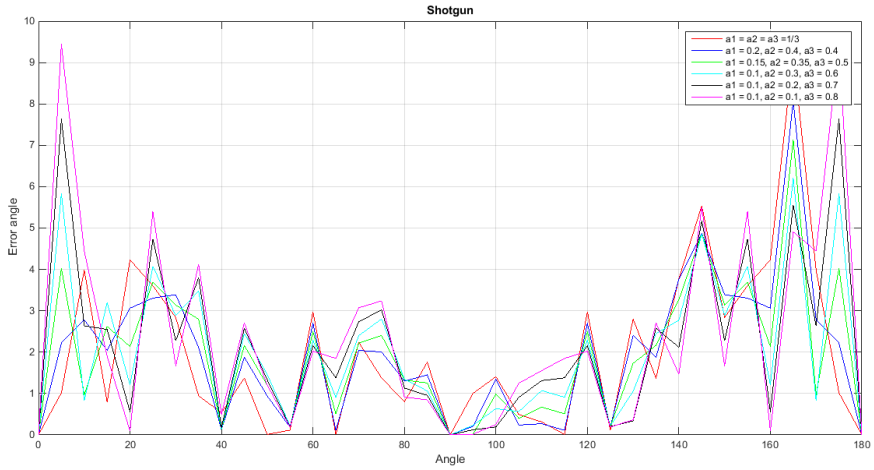


Figura 4.2: Esempi degli errori di stima dell'angolazione. In questo caso si considera come evento sonoro l'urlo di una persona spaventata e si sono applicati pesi uguali (quindi uguali ad 1/3) a tutte le coppie.

In ulteriori esperimenti si sono analizzati i risultati prodotti dall'algortimo variando i valori dei pesi relativi alle coppie di microfoni. Questi risultati sono riportati nella Fig.4.3. Innanzitutto si nota come gli errori commessi nel caso dello shotgun risultano essere leggermente minori rispetto alle urla, probabilmente a causa della natura impulsiva del primo suono che permette di individuare con maggiore precisione i TDOA. Per quanto riguarda la variazione dei pesi, sono stati effettuati alcuni test partendo dall'assegnare pesi uguali a tutte le coppie (1/3) ed assegnando sempre più peso alla coppia (1,4) e meno peso alle restanti coppie. Tali test hanno evidenziato come non esista una differenza significativa nella variazione, se non nel fatto che aumentare l'importanza della coppia (1,4) porti a commettere più errori nel range d'angoli [0°, 90°] e meno errori nel range [90°, 180°]. La tabella 4.2 presenta gli errori medi assoulti compiuti dall'algoritmo al variare della sorgente sonora e dei pesi utilizzati. Viene confermato il fatto che in presenza dello shotgun l'algoritmo commetta meno errori rispetto allo scream e che non ci sia pressoché differenza nell'utilizzare pesi diversi o pesi uguali. Ciò consiglia di tenere pesi uguali per le coppie di microfoni, visto anche che in questo modo un possibile errore compiuto da una coppia di microfoni può essere compensato dalle misurazioni delle altre coppie. Infine, considerando un range ridotto di [30°, 150°], si nota come la precisione aumenti di circa 1-2°. Ciò conferma quanto descritto nell'SDK di Microsoft per quanto riguarda il calcolo dell'angolazione della sorgente sonora, dove consigliano come range del campo di ricezione audio l'intervallo [40°, 140°].



(a) Scream



(b) Shotgun

Figura 4.3: Errore assoluto compiuto nella stima dell'angolazione variando i pesi assegnati alle coppie di microfono e il tipo di sorgente sonora.

Pesi asseganti	Scream	Shotgun	Scream (optimal range)	Shotgun (optimal range)
1/3, 1/3, 1/3	3.71°	1.98°	3.09°	1.59°
0.2, 0.4, 0.4	3.26°	1.96°	3.00°	1.65°
0.15, 0.35, 0.5	3.22°	1.94°	2.97°	1.69°
0.1, 0.3, 0.6	3.35°	2.02°	3.10°	1.75°
0.1, 0.2, 0.7	3.48°	2.16°	3.16°	1.74°
0.1, 0.1, 0.8	3.67°	2.34°	3.24°	1.79°

Tabella 4.2: Errori medi assoluti compiuti variando i pesi assegnati e il tipo di sorgente sonora. Come optimal range si intende [30°, 150°].

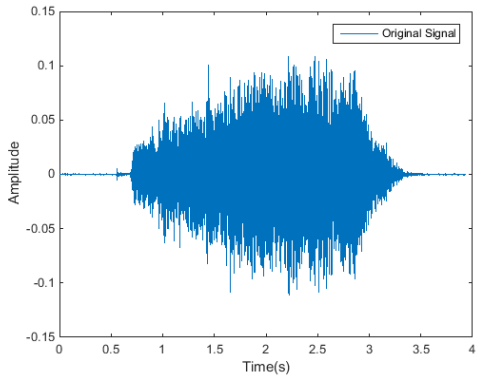
Infine, tenendo conto della limitazione hardware che il dispositivo Kinect soffre, ottenere stime di angolazioni che in media differiscono di circa 2-3° gradi risulta essere piuttosto positivo.

4.1.5 Risultati su angolazioni in presenza di AWGN

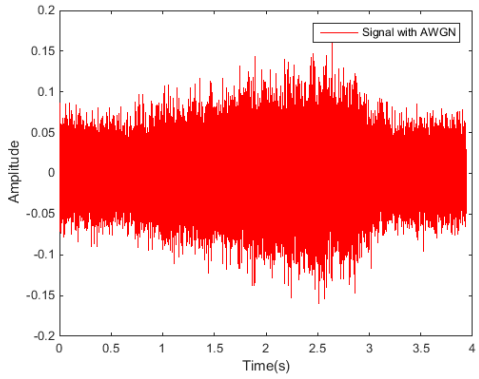
Come ultimo test riguardo la localizzazione sonora, ad ogni file audio registrato è stato aggiunto un rumore bianco (AWGN) in modo da verificare le prestazioni dell'algoritmo in condizioni non ottimali di registrazione sonora. Per diversificare l'intensità del rumore bianco, si è intervenuti sul livello di SNR del segnale audio. Si ricorda che

$$SNR = \frac{Potenza\ segnale}{Potenza\ rumore}$$

Nella Fig.4.4 viene mostrato un esempio di una registrazione audio di uno scream con l'aggiunta del rumore bianco.



(a) Suono originale



(b) Suono affetto da rumore bianco

Figura 4.4: Esempio di un suono (in questo caso uno scream) affetto da rumore bianco

SNR	Scream	Shotgun
-10 db	5.80°	6.68°
-5 db	3.89°	4.37°
0 db	3.97°	3.64°
5 db	3.71°	3.71°
10 db	3.71°	3.71°

Tabella 4.3: Errori medi assoluti compiuti variando il livello di SNR.

Come livelli di SNR si sono scelti valori non elevati, poiché l'intenzione era di verificare l'efficacia dell'algoritmo in condizioni di notevole presenza di rumore. Analizzando i risultati di angolazione ottenuti (sono descritti nella Tabella 4.3) si può notare come, rispetto alla Tabella 4.2, soprattutto con bassi livelli di SNR gli errori di angolazione tendono a crescere di circa 1-2°, mentre con livelli di SNR superiori gli errori tendono a diminuire e ad assestarsi su valori molto simili alle stime in assenza di rumore bianco. Si intende precisare che difficilmente gli eventi sonori scelti possano essere riprodotti nella realtà con livelli di SNR $\leq$ 0db o comunque così bassi, ma nella evenienza di tale fatto si può comunque contare su una buona stima dell'angolazione della sorgente sonora.

4.2 Riconoscimento sonoro

4.2.1 Setup dell'esperimento

Per quanto riguarda il dataset di partenza, esso è stato creato tramite una raccolta di file audio su vari siti Web specializzati nella condivisione audio, liberi da qualsiasi diritto di copyright. Nel caso degli scream, la maggior parte dei campioni proviene da registrazioni di film dell'orrore mentre una parte minoritaria da riproduzioni di urla condivise nei siti specializziati. Nel caso dei shotgun, in tutti i casi tali campioni provengono da riproduzioni di colpi di vari tipi di armi da fuoco in ambiente controllato e anch'essi condivisi online. Infine, per coprire il caso dei suoni non appartenenti alle due precedenti categorie, si sono scelti suoni completamente casuali. In tutto si sono raccolti 120 file audio in formato WAV, equamente suddivisi nelle tre categorie descritte. Si ricorda che, scelto un evento sonoro tra le prime due categorie, i campioni che caratterizzano i suoni non appartenenti a tale evento sono l'unione dei suoni random e dell'evento sonoro non considerato. Perciò indipendentemente dall'evento sonoro scelto, si ha a disposizione un numero di 40 campioni positivi (corrispondenti al tipo scelto) e 80 campioni negativi per il dataset.
I test di machine learning sono stati compiuti su tale dataset utilizzando i modelli descritti nel Capitolo 3 e tramite il metodo di validazione 10-cross-validation. In particolare nell'AdaBoostM1 il classificatore di base scelto è stato il C4.5. Di seguito vengono rammentate le feature audio scelte:

- zero-crossing rate;
- centroide spettrale;
- spectral flatness;
- spectral skewness;
- l'attack slope;
- il decrease slope.
- i primi 13 MFCC;

4.2.2 Risultati del 10-cross-validation

Nelle seguenti tabelle verranno mostrati i risultati prodotti in diversi test. Come metriche di valutazione per le prestazioni dei modelli di Machine Learing si sono usate le seguenti:

- percentuale di correttezza delle istanze;
- true positive (TP), il numero di istanze positive correttamente predette;
- false negative (FN), il numero di istanze negative scorrettamente predette;
- false positive (FP), il numero di istanze positive scorrettamente predette;
- true negative (TN), il numero di istanze negative correttamente predette;
- sensitivity, uguale a $TP/(TP + FN)$, la frazione di istanze positive correttamente predette;
- specificity, uguale a $TN/(TN + FP)$, la frazione di istanze negative correttamente predette;
- precision, uguale a $TP/(TP + FP)$, la frazione di istanze predette positive che effettivamente risultano positive.

Nella Tabella 4.4 vengono elencati i risultati ottenuti con l'evento sonoro scream. I modelli C4.5 e AdaBoostM1 (che ha come classificatore base proprio il C4.5) hanno mostrato le prestazioni peggiori, con un valore di precisione di C4.5 a 0.65. Tuttavia, come era prevedibile, AdaBoostM1 si è comportato meglio rispetto a C4.5. GMM e AODE presentano valori di correttezza molto simili, intorno al 90%, ma AODE ha un valore di sensitività migliore rispetto a GMM (0,925 contro 0,825). Il modello migliore è stato il Random Forest: correttezza del 96,7%, sensitività di 0,925, specificità di 0,988 e precisione di 0,974. In generale la specificità è sempre stata piuttosto alta in tutti i modelli considerati, dimostrando quindi una buona attitudine a predire correttamente i suoni non scream, mentre si possono promuovere i modelli AODE, e soprattutto Random Forest, nel predire lo scream.

Model	% corrected	TP	FN	FP	TN	Sensitivity	Specificity	Precision
C4.5	82.5	26	14	3	73	0,650	0,913	0,650
Random Forest	96,7	37	3	1	79	0,925	0,988	0,974
AdaBoostM1	86,7	29	11	5	73	0,725	0,938	0,853
AODE	91,7	37	3	7	37	0,925	0,913	0,841
GMM	89,2	33	7	6	74	0,825	0,925	0,846

Tabella 4.4: Risultati dei modelli sullo scream.

Nella Tabella 4.5 vengono elencati i risultati ottenuti con l'evento sonoro shotgun. Ponendoli a confronto con i relativi dello scream, si nota un netto miglioramento complessivo nella correttezza delle predizioni, a parte il caso del GMM dove si nota un leggero peggioramento. Partendo da tale modello, esso non commette alcun errore a predire le istanze appartenenti ai random: nessun FP e specificità e precisione massime. D'altro canto non si può dire lo stesso per quanto riguarda la label shotgun: 14 FN (quasi la metà) e sensitività di 0,65. Considerando gli altri modelli, se C4.5 si avvicina come correttezza al GMM (nonostante sia più equilibrato nelle prestazioni con le differenti label) Random Forest, AdaBoostM1 e AODE presentano valori molto positivi: correttezza del 95% o superiore, specificità e precisione intorno al 0,95.

Model	% corrected	TP	FN	FP	TN	Sensitivity	Specificity	Precision
C4.5	89,2	33	7	6	74	0,825	0,925	0,846
Random Forest	97,5	39	1	2	78	0,975	0,975	0,951
AdaBoostM1	97,5	39	1	2	78	0,975	0,975	0,951
AODE	95,0	36	4	2	78	0,900	0,975	0,940
GMM	88,3	26	14	0	80	0,650	1,000	1,000

Tabella 4.5: Risultati dei modelli sullo shotgun.

Dopo aver illustrato le due precedenti tabelle e considerando i risultati combinati, il modello migliore è stato il Random Forest: in entrambi i casi i valori di correttezza, sensitività e precisione sono state sempre molto alti (0,95 o superiori). Un'alternativa al Random Forest è l'AODE, che ha mostrato prestazioni molto simili al Random Forest. Si sconsiglia invece l'utilizzo di C4.5 che nel complessivo è stato il peggiore. Un ulteriore test, tramite il software Weka, è stato operare alcuni algoritmi di feature selection su entrambi i dataset. Tali algoritmi hanno lo scopo di selezionare dall'intero insieme di attributi un sottoinsieme di questi. In tal modo solamente i più rilevanti vengono considerati e le prestazioni del modello di Machine Learning utilizzato possono subire miglioramenti. In particolare sono stati utilizzati i seguenti:

- *ChiSquaredAttributeEval*, che valuta l'importanza di un attributo calcolando il valore della statistica chi-quadro rispetto alla label;
- *GainRatioAttributeEval*, che utilizza come metro di confronto tra attributi il gain ratio;

- *InfoGainAttributeEval*, che utilizza l'information gain;
- *CorrelationAttributeEval*, che calcola la correlazione di Pearson tra l'attributo e la label.

Nel lavoro di tesi il procedimento è stato unire tali algoritmi con la tecnica del Sequential Bacward selection (SBS). Tale tecnica consiste nel partire con l'intero set di attributi e, eliminando un attributo alla volta, verificare i cambiamenti di prestazioni. Applicando tale strategia, si è giunti a selezionare come attributi un sottoinsieme dei MFCC, in particolare i coefficienti $[3, 5, 8, 9, 10, 11, 12]$. Utilizzando tale sottoinsieme con i rimanenti modelli di predizione si sono ottenuti i risultati elencati nelle Tabelle 4.6-4.7.

Model	% corrected	TP	FN	FP	TN	Sensitivity	Specificity	Precision
C4.5	85,8	29	11	6	74	0,725	0,925	0,829
Random Forest	93,3	35	5	3	77	0,875	0,963	0,921
AdaBoostM1	92,5	35	5	4	76	0,875	0,950	0,897
AODE	90,0	35	5	7	73	0,875	0,913	0,833
GMM	86,7	32	8	8	72	0,800	0,900	0,900

Tabella 4.6: Risultati dei modelli sullo scream applicando la feature selection.

Nella Tabella 4.6 vengono elencati i risultati ottenuti con l'evento sonoro scream applicando la feature selection. Confrontando i risultati ottenuti con la Tabella 4.4, a parte il caso di C4.5 che ha migliorato la percentuale di correttezza ma è rimasto comunque il peggiore, la selezione non ha portato a miglioramenti poiché i risultati sono pressoché simili. I GMM hanno addirittura mostrato prestazioni inferiori. Dopo tali considerazioni si evince che per il caso dello scream sia produttivo lasciare interamente le feature selezionate inizialmente.

Model	% corrected	TP	FN	FP	TN	Sensitivity	Specificity	Precision
C4.5	93,3	36	4	4	76	0,900	0,95	0,900
Random Forest	95,8	38	2	3	77	0,950	0,963	0,925
AdaBoostM1	94,2	37	3	4	76	0,925	0,95	0,900
AODE	92,5	36	4	5	75	0,900	0,938	0,878
GMM	93,3	32	8	0	80	0,800	1,000	1,000

Tabella 4.7: Risultati dei modelli sullo shotgun applicando la feature selection.

Nella Tabella 4.7 vengono elencati i risultati ottenuti con l'evento sonoro shotgun applicando la feature selection. Al contrario della Tabella 4.6, si possono notare in alcuni casi dei notevoli miglioramenti. Per esempio GMM ha migliorato la sua percentuale di correttezza da 88,3% a 93,3% e, se anche in questo caso non commette

errori a classificare le istanze negative, ha migliorato la sua predizione delle istanze positive. Ciò nonostante la sua sensitività non supera l'80%. Allo stesso modo si notano miglioramenti nel modello C4.5 che presenta valori molto simili nei confronti di AdaBoostM1, che si ricorda utilizza proprio C4.5 come classificatore base. Invece le prestazioni dei modelli Random Forest, AdaBoostM1 e AODE sono stati leggermente inferiori rispetto alla Tabella 4.5, ma si assestano su valori molto promettenti (correttezza del 93-95%, sensitività del 0,9-0,95 e precisione intorno al 0,9). Anche in questo caso il migliore modello nel predirre entrambi i tipi di istanze è stato il Random Forest.
Dopo aver analizzato i risultati mostrati precedentemente, si sono ottenute le seguenti conclusioni:

- In generale i modelli hanno predetto nel complessivo con maggiore precisione l'evento sonoro shotgun rispetto all'evento scream.
- Nel caso dello scream, si consiglia di tenere le feature scelte inizialmente e di utilizzare come modello di predizione il Random Forest.
- Nel caso dello shotgun, le prestazioni con e senza feature selection sono state pressoché simili nel caso del Random Forest, mentre nei modelli con prestazioni inferiori si sono evidenziati miglioramenti dopo l'utilizzo della feature selection. Comunque anche in questo caso, visto che il numero di feature iniziale (19) non si può considerare elevato, si consiglia di non applicare tale selezione e di utilizzare Random Forest.

4.2.3 Confronto con risultati simili dello stato dell'arte

Un utile confronto è osservare tali risultati in relazione a quelli presenti in [11]. Anch'esso tratta la stessa tematica di riconoscimento sonoro dei scream e dei shotgun, ma utilizza come modello di predizione il GMM unito all'algoritmo Figuereido e Jain (FJ). [11] ha utilizzato lo stesso procedimento di raccolta campioni dei due eventi sonori (registrazioni da film e librerie audio online), mentre per quanto riguarda i suoni non appartententi alle due categorie sono stati registrati campioni audio in un'area pubblica. Infine ai campioni dei due eventi sonori è stato aggiunto un rumore bianco con un determinato livello di SNR.
Tale confronto è presente nella Tabella 4.8, dove le metriche di confronto sono la precision e il recall, cioè la sensitività. Come si può notare, il valore di precisione di GMM-EM, il modello implementato in questa tesi, risulta essere leggermente a favore rispetto a GMM-FJ, mentre GMM-FJ si comporta decisamente meglio rispetto a GMM-EM nella metrica recall. Infatti l'algoritmo FJ è stato sviluppato per poter sopperire alle limitazioni presenti nell'algoritmo EM. Invece nel confronto con il Random Forest, quest'ultimo presenta valori di precisione e di recall superiori, dimostrando l'efficacia di tale modello.

Model	Precision	Recall
GMM-EM	0,923	0,737
Random Forest	0,963	0,950
GMM-FJ	0,900	0,920

Tabella 4.8: Confronto tra i risultati ottenuti in questa tesi e quelli contenuti in [11]. Il modello utilizzato da [11] viene per chiarezza denominato GMM-FJ. Per ottenere i valori di precisione e di precision/recall, è stata applicata la media tra i risultati delle tabelle precedenti.

Si precisa che il livello di SNR indicato da [11] per i risultati elencati è di 10-15dB, perciò non elevato e in linea con le caratteristiche dei campioni raccolti nel lavoro di tesi, cioè in molti casi affetti da rumore. Il fatto che i valori di GMM-EM risultino coerenti con i relativi di GMM-FJ lo dimostra. Nonostante siano stati utilizzati due dataset differenti, l'aver adoperato lo stesso metodo di raccolta campioni e la varietà utilizzata nel comporre il dataset analizzato dai modelli giustifica il confronto esaminato precedentemente.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Conclusioni

In questo lavoro di tesi è stata proposta una implementazione della localizzazione sonora di una sorgente sonora tramite il dispositivo Kinect v1 e del riconoscimento di due eventi sonori, scream e shotgun.
Nel caso della localizzazione (angolazione della sorgente sonora) gli errori sono stati intorno ai 2° per il caso dello shotgun e di 3° per lo scream. Considerando le relative problematiche legate al dispositivo Kinect utilizzato (quantizzazione dovuta a sample rate e bassa apertura degli array) i risultati sono stati positivi, visto anche l'obiettivo con cui il dispositivo è stato concepito (localizzare il range di provenienza del suono). Inoltre si è mostrato come la predizione dell'angolazione sia robusta alla presenza di rumore bianco, anche di forte intensità.
Per quanto riguarda il riconoscimento sonoro, è stata implementata una feature extraction su campioni di file audio raccolti e sono stati utilizzati alcuni noti modelli di Machine Learning. Dai risultati si è concluso che il miglior modello da utilizzare è il Random Forest, poiché possiede valori di correttezza intorno al 96-97%, e valori intorno al 95% di sensitività, specificità e precisione.
Tali risultati evidenziano che, nonostante l'utilizzo di microfoni integrati a basso costo, sia possibile ottenere prestazioni paragonabili allo stato dell'arte, se non in alcuni casi superiori, nel campo della localizzazione e del riconoscimento sonoro.

5.2 Sviluppi futuri

Uno dei possibili sviluppi futuri consiste nel poter usurfruire in ambiente opensource della versione successiva al Kinect v1, il v2. Infatti al momento della stesura di questa tesi una libreria opensource con tali caratteristiche non è disponibile. Il Kinect v2, nonostante abbia le stesse caratteristiche hardware della componente sonora rispetto al modello v1, ha una componente video molto più aggiornata: il v2 presenta una camera con risoluzione 1080p in confronto alla risoluzione VGA del v2. Tale fattore, nel contesto dell'audio-video sorveglianza, può migliorare la registrazione video dell'evento sonoro. OpenKinect al momento ha sviluppato una libreria in grado di utilizzare la compomente video del v2, *libfreenect2*, ma non la componente audio. Inoltre, in un ulteriore perfezionamento del sistema di localizzazione-riconoscimento

sonoro presentato in questo elaborato, la registrazione di campioni reali e un numero maggiore di questi migliorerebbero la costruzione del modello di riconoscimento, potendo avere a disposizione un dataset più ampio e variegato.

Bibliografia

[1] Wireless Broadband Alliance Industry Report 2013: Global Trends in Public Wi-Fi. `http://www.seagate.com/files/www-content/solutions-content/surveillance-security-video-analytics/en-us/docs/video-surveillance-trends-report.pdf`, 1 Sept. 2016.

[2] Brandstein and D. Ward (Eds). *Microphone Arrays*, Springer, 2001.

[3] Sito ufficiale del prodotto Kinect v1, `http://www.xbox.com/en-US/xbox-360/accessories/kinect`, 1 Sept. 2016.

[4] Zhengyou Zhang, *Microsoft Kinect Sensor and Its Effect*, IEEE MultiMedia, April 1 2012, Vol.19, num.2, pages 4-12.

[5] Cha Zhang, Dinei Florêncio, Demba E. Ba, Zhengyou Zhang *Maximum Likelihood Sound Source Localization and Beamforming for Directional Microphone Arrays in Distributed Meetings*, IEEE Transactions on Multimedia, April 2008, Vol. 10, num. 3, pages 538-548.

[6] Sito ufficiale di Windows con le specifiche tecniche di Kinect v1, `https://msdn.microsoft.com/en-us/library/jj131033.aspx`, 1 Sept. 2016.

[7] Sito ufficiale dell'SDK di Windows per Kinect, `https://developer.microsoft.com/it-it/windows/kinect`, 1 Sept. 2016.

[8] A. Mahajan and M. Walworth. *3-d position sensing using the difference in the time-of-flights from a wave source to various receivers*, IEEE Transactions on Robotics and Automation, 17(1), pages 91-94, 2001.

[9] J.M. Varin, F.Michaud, J.Rouat, D.Letourneau, *Robust sound source localization using a microphone array on a mobile robot*, IROS 2003, Vol.2, pages 1228-1233, Oct. 2003.

[10] Lawrence R. Rabiner, B. H. Juang, *Fundamentals of speech recognition*, Prentice Hall, 1° Edition, April 1993.

[11] L.Gerosa, G.Valenzise, M.Tagliasacchi, F.Antonacci, A.Sarti *Scream and gunshot detection in noisy environments*, Signal Processing Conference, pages 1216 - 1220, 3-7 Sept. 2007.

[12] J. D. Johnston, *Transform coding of audio signals using perceptual noise criteria*, IEEE Journal on Selected Areas in Communications, Vol.6, Issue 2 , pages 314 - 323, Feb 1988.

[13] S. S. Stevens, J. Volkmann, E. B. Newman, *A Scale for the Measurement of the Psychological Magnitude Pitch*, The Journal of the Acoustical Society of America, Vol. 8, Issue 3, page 185, 1937.

[14] Giovanni De Poli, Luca Mion, Nicola Orio, *Algorithms for sound and music computing*, Università di Padova, Chapter 6, pages 1-28, March 12 2012.

[15] C. F. Eick, N. Zeidat, Z. Zhao, *Supervised clustering - algorithms and benefits*, Tools with Artificial Intelligence, pages 774 - 776, 15-17 Nov. 2004.

[16] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan-Kaufmann Publishers, San Mateo, CA, 1993.

[17] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg, *Top 10 algorithms in data mining*, Knowledge and Information Systems, Vol.14, No.1, pages 1-37, 2008.

[18] Leo Breiman. Random Forest. Machine Learning, 45, 5-32, 2001 Kluwer Academic Publishers.

[19] Leo Breiman, Adele Cutler. Random Forests. `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_papers.htm`, 1 Sept. 2016.

[20] Yoav Freund, Robert E. Schapire. *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences, Vol.55, Issue 1, pages 119-139, Aug. 1997

[21] Yoav Freund, Robert E. Schapire. *Experiments with a New Boosting Algorithm*, Journal of Computer and System Sciences, Proceedings of the Thirteenth International Conference on Machine Learning, Jan. 1996

[22] Webb, I. Geoffrey, Boughton, R. Janice, Wang, Zhihai,*Not so naive Bayes: Aggregating one-dependence estimators* , Machine Learning, Vol.5, No.1, pages 5-24, 2005

[23] McLachlan, G., and D. Peel, *Finite Mixture Models* , NJ: John Wiley & Sons, Inc., 2000.

[24] A. P. Dempster, N. M. Laird and D. B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1, pages 1-38, 1977.

[25] Olivier Lartillot, Petri Toiviainen, *A Matlab Toolbox for Musical Feature Extraction From Audio*, International Conference on Digital Audio Effects, Bordeaux, 2007.