ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA

Scuola di Ingegneria ed Architettura

Corso di Laurea in Ingegneria dei Sistemi Elettronici per lo Sviluppo Sostenibile

IMPLEMENTAZIONE DI ALGORITMI DI LOCALIZZAZIONE DI SORGENTI AUDIO

Elaborato in
Laboratorio di Elettronica e Telecomunicazioni

Tesi di Laurea di:
ENRICO TESTI

Relatore:
Prof. Ing.
ANDREA GIORGETTI

SESSIONE II
ANNO ACCADEMICO 2014–2015

PAROLE CHIAVE

Localizzazione

Stima angolo di arrivo

Elaborazione audio

MATLAB

Simulazione

Indice

Introduzione

Per *sound location* si intende la capacità di determinare la direzione e la distanza di una sorgente con il solo aiuto del suono da essa emesso. [1]

Il calcolo di questi due parametri può, ad esempio, permettere un'accurata localizzazione di uno speaker; esistono infatti sistemi low-cost basati su tale concetto ed in grado di operare con elevata precisione. Alcune applicazioni richiedono la sola stima della direzione di arrivo di un suono; ad esempio durante una videoconferenza il sistema potrebbe individuare la direzione di provenienza della voce dell'utente e direzionare lo speaker in modo tale da migliorare l'esperienza di ascolto.

Per un sistema antifurto di un'abitazione potrebbe invece essere utile la stima della posizione della sorgente sonora, permettendo così la localizzazione di eventuali intrusi.

Questo progetto è stato ideato con lo scopo di implementare e simulare un algoritmo di stima della posizione di una sorgente sonora. Per comodità ed efficienza dell'applicazione è stato scelto di utilizzare MATLAB come strumento di sviluppo. Tutti i dispositivi utilizzati sono compatibili, interfacciabili tra loro ed ampiamente descritti all'interno di questo elaborato.

Ad un'accurata descrizione dell'algoritmo utilizzato, della sua implementazione in MATLAB, dei dispositivi impiegati e del loro interfacciamento segue un capitolo dedicato alla progettazione di un'interfaccia grafica in grado di semplificare notevolmente la simulazione e permettere, anche ad un utente non dotato di particolari competenze attinenti al linguaggio MATLAB, di effettuare un test dell'algoritmo stesso.

Capitolo 1

Algoritmo di Localizzazione

Il problema della localizzazione di sorgenti di onde in generale, audio in questo caso, è stato affrontato in modalità differenti nel corso della storia. In questo trattato si è deciso di suddividere il suddetto in due sottoproblemi:

- Stima della *Direction of Arrival*(*DoA*) o *Direzione di Arrivo*.
- *Triangolazione* della posizione della sorgente.

1.1 Direction of Arrival

È noto come l'uomo sia in grado di stimare la direzione di provenienza di un suono; il sistema formato da orecchie e cervello può infatti, senza ricorrere ad alcun tipo di aiuto esterno, ricevuto un segnale, elaborarlo e dedurne approssimativamente la posizione della sorgente. Questo è possibile grazie alla particolare forma dell'orecchio umano ed al ritardo causato dalla propagazione del suono. [2]

Nell'elaborazione dei segnali la *Direction of Arrival* è definita come direzione con la quale un'onda giunge in un punto ben definito, dove solitamente sono collocati un determinato numero di sensori o microfoni, in questo caso specifico.

In letteratura sono presenti algoritmi differenti per la stima della direzione di arrivo; per questo progetto è stato ideato un algoritmo funzionante mediante l'ausilio di due soli microfoni. L'algoritmo in questione si basa sulla stima del *delay* o ritardo tra i segnali ricevuti dai due microfoni attraverso il calcolo della funzione di Cross-Correlazione.

Una volta noto il ritardo è possibile ottenere la direzione di arrivo attraverso l'utilizzo della trigonometria.

1.1.1 Cross-Correlazione

Nella teoria dei segnali la funzione di *Cross-Correlazione* o *Correlazione Incrociata* rappresenta la somiglianza tra due segnali valutata in base ad uno spostamento o traslazione temporale applicata ad uno di essi.

Per due segnali ad *energia finita* la funzione di Cross-Correlazione è definita infatti come: [3]

$$R_{xy}(t) = \int_{-\infty}^{+\infty} x^*(\tau)y(t+\tau)d\tau \tag{1.1}$$

o nel caso di segnali tempo-discreti:

$$R_{xy}[n] = \sum_{m=-\infty}^{+\infty} x^*(m)y(n+m) \tag{1.2}$$

Per due segnali a *potenza finita* è invece definita come:

$$R_{xy}(t) = \lim_{T\to\infty} \frac{1}{T} \int_{0}^{T} x^*(\tau)y(t+\tau)d\tau \tag{1.3}$$

MATLAB fornisce già uno strumento per il calcolo di questa funzione[4], tuttavia si è preferito realizzare a parte un codice in grado di replicarlo con elevata precisione. Dati *x1[n]* e *x2[n]*, vettori contenenti i campioni dei due segnali in ingresso:

- Inizialmente viene calcolata la lunghezza del vettore che conterrà il risultato dell'operazione, considerando le rispettive lunghezze dei singoli segnali. Tale valore viene poi approssimato alla potenza di 2 più vicina.

  ```
  lungcorr=length(x1)+length(x2)-1;
  N=pow2(nextpow2(lungcorr));
  ```

- Successivamente viene solto il calcolo della funzione di Cross-Correlazione e memorizzato il risultato in un vettore *z[m]*. I comandi *Tic* e *Toc*

permettono di monitorare il tempo impiegato dal sistema ad eseguire l'operazione.

```
tic
z=real(ifft(fft(x1,N).*fft(x2(end:-1:1),N)));
toc
```

- Per verificare poi il risultato ottenuto viene effettuato un plot del vettore *z[m]*. Gli assi x e y sono generati a partire dalla frequenza di campionamento e dalla lunghezza del vettore precedentemente calcolata.

```
dt=1/fs;
timeaxis=[-lungcorr/2+1:1:lungcorr/2]*dt;
plot(timeaxis,z(1:length(timeaxis))),xlabel('time lag [s]'),...
ylabel('Correlation'),legend('Corr(x_n,y_n)');
```

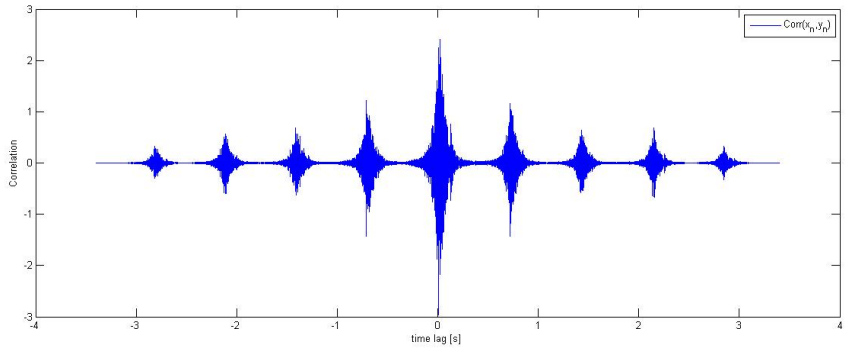

Figura 1.1: Esempio di funzione di Cross-Correlazione

1.1.2 Calcolo del ritardo

Una volta svolto il calcolo della funzione di Cross-Correlazione è possibile procedere con la stima del ritardo dovuto alla propagazione dei due segnali nello spazio libero:

- Prima di tutto è necessario localizzare il picco della funzione di Cross-Correlazione. Il vettore *locmax* contiene la posizione del suddetto, mentre *maxval* contiene il suo valore in modulo.

```
[maxval,locmax]=max(abs(z));
```

- Ora è possibile calcolare il ritardo espresso in campioni sottraendo alla posizione del picco la lunghezza del segnale *x1[n]*. Per ottenere il ritardo temporale è sufficiente moltiplicare il risultato ottenuto per *dt* (reciproco della frequenza di campionamento *fs*).

```
delay=(locmax-length(x1))*dt;
```

1.1.3 Trigonometria

Ottenuto il valore del ritardo temporale tra i due segnali puó essere stimata la direzione di arrivo. Prima di tutto è necessario definire un sistema di riferimento xOy all'interno del quale posizionare i due microfoni A e B ad una distanza d. Il sistema scelto (Figura 1.2) è tale per cui l'asse y è asse di simmetria del segmento AB. L'obbiettivo è l'angolo α individuato dall'asse y e dalla retta congiungente la sorgente M con il punto medio O di AB.

Ricordando la velocità di propagazione del suono v = 340.29 m/sec al livello del mare ed alla temperatura di $20°C$[5] si ha che, detto τ il ritardo precedentemente calcolato:

$$AB' = \tau v \tag{1.4}$$

Utilizzando la funzione *arcsin* è quindi possibile in questo modo ottenere l'angolo desiderato in radianti:

```
alpha=asin(v*delay/d);
```

Segue poi la conversione in gradi:

```
alpha=alpha/pi*180;
```

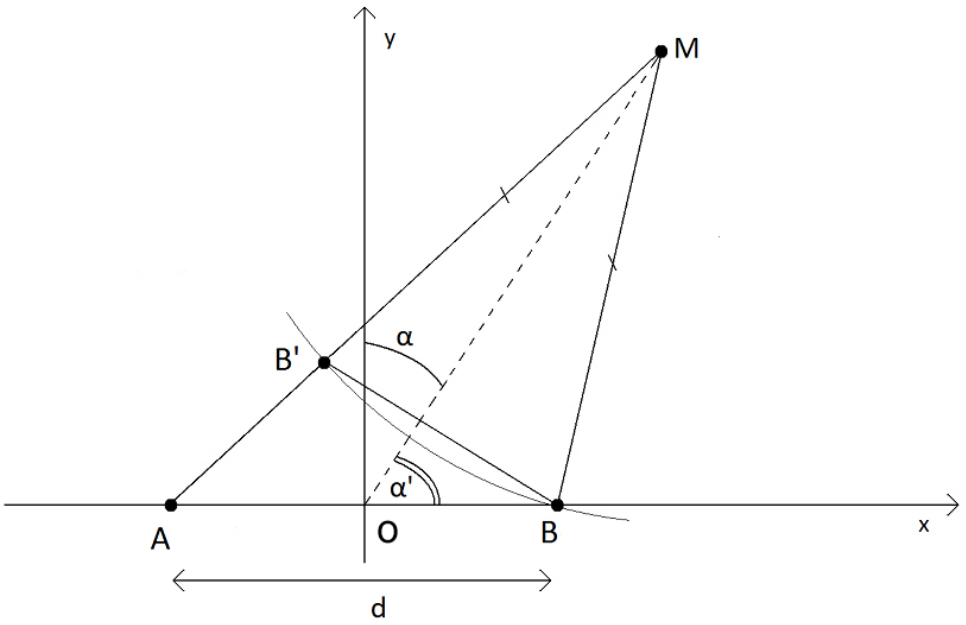

Figura 1.2: Sistema di Riferimento

1.2 Triangolazione

Il passo successivo alla stima della direzione di arrivo è il calcolo delle coordinate della posizione della sorgente. Per la realizzazione di questa fase è necessario svolgere il procedimento descritto precedentemente (Capitolo 1.1) utilizzando due coppie di microfoni, posizionate in modo tale da fornire due misure differenti di angolo di arrivo del segnale.

Il procedimento consiste nel calcolo delle coordinate del punto di intersezione delle due rette corrispondenti alle direzioni di arrivo ottenute rispettivamente dalle due misurazioni. Per comodità e per ottenere un'ampia copertura si è scelto di posizionare i quattro microfoni all'interno del sistema di riferimento come in Figura 1.3.

La retta identificata dalla direzione di arrivo individuata dai microfoni A e B ha un'equazione del tipo:

$$y = mx \tag{1.5}$$

Mentre quella individuata dai microfoni C e D:

$$(y - y_{O'}) = m(x - x_{O'}) \tag{1.6}$$

$$y = mx - mx_{O'} + y_{O'} \tag{1.7}$$

Dove $x_{O'}$ e $y_{O'}$ sono le coordinate del punto medio del segmento CD, mentre m é la tangente dell'angolo α per la prima retta e dell'angolo $180° - \beta$ per la seconda.

Il sistema da risolvere per il calcolo della posizione è quindi:

$$\begin{cases} y = m_1 x \\ y = m_2 x - m_2 x_{O'} + y_{O'} \end{cases} \tag{1.8}$$

Le cui soluzioni sono:

$$\begin{cases} x_M = \frac{y_{O'} - m_2 x_{O'}}{m_1 - m_2} \\ y_M = m_1 \frac{y_{O'} - m_2 x_{O'}}{m_1 - m_2} \end{cases} \tag{1.9}$$

La sorgente audio così individuata si troverà quindi alle coordinate x_M e y_M del sistema di riferimento scelto.

Su MATLAB il procedimento è analogo e può essere realizzato come segue:

```
m1=tand(alpha)
m2=tand(180-beta)
Xp = (yc - m2*xc)/(m1 - m2);
Yp = m1*(yc - m2*xc)/(m1 - m2);
```

Per verificarne il funzionamento è sufficiente l'utilizzo della funzione *plot*, generando accuratamente gli assi e posizionando i quattro microfoni. La distanza tra i microfoni A e B è detta *d1* mentre quella tra C e D è detta *d2*:

```
assey1=assex*m1;
assey2=assex*m2 - xc*m2 + yc;
plot(assex,assey1,assex,assey2),axis([-6 6 0 6]) ;
hold on;
plot([xc xc],[yc+d2/2 yc+d2/2],'o');
plot([xc xc],[yc-d2/2 yc-d2/2],'o');
plot([d1/2 d1/2],[0 0],'o');
plot([-d1/2 -d1/2],[0 0],'o');
plot([Xp Xp],[Yp Yp],'bd');
```

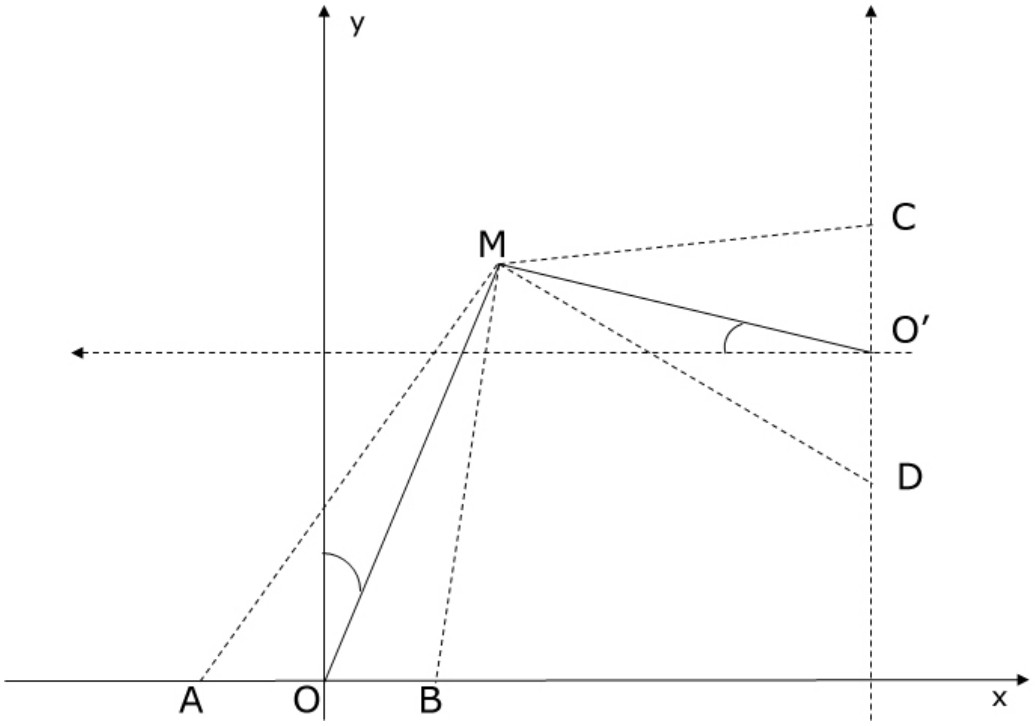

Figura 1.3: Schema di riferimento

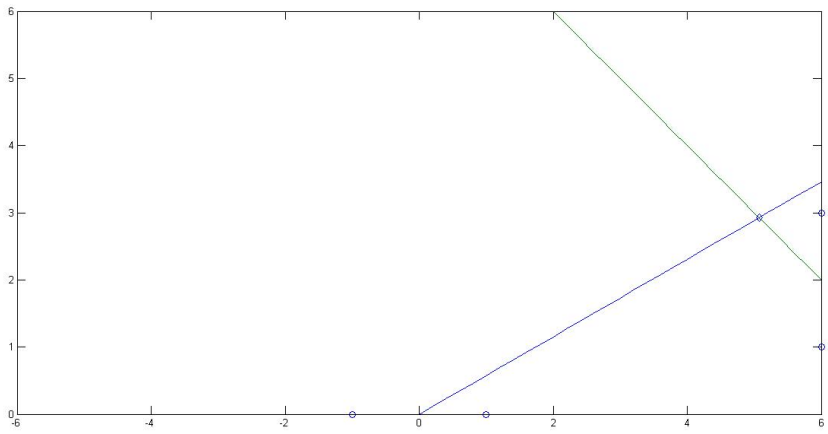

Figura 1.4: Esempio di localizzazione di una sorgente audio. Sono riportate anche le posizioni dei 4 microfoni, identificate dai cerchi blu.

Capitolo 2

Scelta Hardware

Dopo aver definito passo a passo gli algoritmi utilizzati è necessario dedicare il giusto spazio alla scelta della strumentazione più consona al raggiungimento dell'obbiettivo del progetto. I parametri valutati durante la scelta dei dispositivi sono:

Numero di canali analogici in ingresso: Per l'utilizzo dell'algoritmo per la stima della posizione di una sorgente audio sono necessari almeno 4 canali analogici in ingresso a cui collegare i 4 microfoni. Con un numero superiore di canali è possibile effettuare più misurazioni, aumentando così la precisione della stima.

Costo: Non sono necessarie soluzioni a costo elevato per il semplice test degli algoritmi.

Latenza di accesso: È fondamentale, nel caso venga utilizzato più di un dispositivo in parallelo, sincronizzarli ed eliminare quindi qualsiasi tipo di ritardo in fase di registrazione. Questo parametro influisce drasticamente sulla precisione dell'algoritmo di stima della posizione.

Le opzioni disponibili per la sperimentazione di questi algoritmi sono innumerevoli; tuttavia sono state vagliate solo le proposte realizzabili in tempo ragionevole. Consideriamo quindi:

- Microphone Arrays
- Digilent Analog Discovery
- Focusrite Saffire Pro 6 USB (Focusrite Scarlett 2i4 USB)

2.1 Microphone Arrays

Per *array di microfoni* si intende un qualsiasi gruppo di microfoni operanti in tandem.[6] Possono essere utilizzati per diversi scopi, tra cui:

- Sistemi per l'estrazione di voci da registrazioni in ambienti particolarmente rumorosi.
- Beamforming
- Registrazioni in alta qualità
- Localizzazione di sorgenti acustiche

Le strutture contenenti i microfoni possono assumere forme differenti a seconda del tipo di impiego per il quale vengono progettati.

In commercio sono presenti array con numeri differenti di dispositivi al loro interno, quindi il numero di canali analogici in ingresso è variabile e deve essere superiore a quattro unità per l' obbiettivo che si vuole raggiungere.

Le strutture sono inoltre progettate in modo da assicurare la perfetta sincronizzazione dei microfoni in fase di registrazione; è quindi possibile ignorare il problema delle latenze.

Si tratta tuttavia di sistemi particolarmente costosi e sovradimensionati rispetto alle specifiche richieste; per questo motivo tale soluzione non è stata designata come migliore per testare gli algoritmi.

Figura 2.1: Esempio di Microphone Array da 24 unità dell'azienda ETS Solutions [7].

2.2 Digilent Analog Discovery

L'*Analog Discovery* è una scheda di acquisizione dati, prodotta dalla Digilent, dotata di numerose funzionalità tra cui [8]:

- Oscilloscopio
- Generatore di forme d'onda
- Analizzatore di spettro
- Voltmetro
- I/O digitali

Per la scheda tecnica completa si rimanda il lettore al sito del costruttore.

L'Analog Discovery è dotata di due canali di ingresso mono analogici a cui possono essere collegati due microfoni; sorge quindi il prolema della mancanza di un numero sufficiente di canali per eseguire il test.

È inoltre progettata per essere totalmente compatibile con MATLAB. Il costo non è particolarmente elevato per studenti e ricercatori; tuttavia neanche questo dispositivo è stato scelto per il test.

Figura 2.2: Digilent Analog Discovery [8]

2.3 Focusrite Saffire Pro 6 USB

La *Sapphire Pro 6* è una scheda audio USB dotata di due canali mono analogici in ingresso e quattro in uscita. I preamplificatori microfonici Focusrite al suo interno garantiscono un'elevata qualità in fase di registrazione. La scheda è dotata di numerose funzionalità tra cui [9]:

- Playback a latenza quasi del tutto nulla durante la registrazione.
- I/O MIDI
- +48Volt per i microfoni che necessitano tensioni di alimentazione elevate.
- Potenziometri di controllo del guadagno dei preamplificatori.

L'interfacciamento con i sistemi operativi più comuni (Windows 7/8/10, Mac OSX, etc..) è estremamente semplice e funzionale: è infatti sufficiente collegarla alla porta USB del computer ed installare i drivers. Una volta avvenuto il riconoscimento della scheda da parte del sistema operativo è possibile accedervi anche tramite MATLAB (Capitolo 3).

Figura 2.3: Focusrite Sapphire Pro 6 USB [9]

Anche in questo caso il costo non risulta particolarmente elevato; tuttavia resta il problema del numero insufficiente di canali analogici in ingresso.

La soluzione è stata ottenuta impiegando due dispositivi in parallelo; collegando infatti due schede audio Focusrite allo stesso computer è stato possibile utilizzare 4 canali mono analogici in ingresso. In questo caso specifico per motivi di comodità e disponibilità è stata impiegata una scheda audio della Focusrite di modello differente ma con le stesse specifiche della precedente. La scheda in questione è la *Scarlett 2i4 USB* [10].

Durante l'utilizzo delle due schede in parallelo sono tuttavia sorti numerosi problemi di sincronizzazione.

Figura 2.4: Focusrite Scarlett 2i4 USB [10]

2.3.1 Problemi di latenza

L'algoritmo ideato per la stima dell'angolo di arrivo confronta il segnale ricevuto da due microfoni e sfrutta lo sfasamento temporale introdotto dalla propagazione del suono nell'ambiente circostante. Per questo motivo la non perfetta sincronizzazione dei due canali utilizzati porta ad un errore nella misurazione.

Collegando un primo microfono ad una scheda, un secondo all'altra ed avviando la registrazione si può immediatamente notare come il tempo impiegato dal computer per accedere alle due schede sia differente. Nonostante i dispositivi siano al loro interno quasi identici e comunichino entrambi con la macchina tramite protocollo USB 2.0, la latenza di accesso ad entrambe è risultata essere di natura puramente aleatoria. Negli studi di registrazione professionali vengono infatti utilizzati apparecchi progettati apposta per sincronizzare tutte le schede audio utilizzate mediante un clock esterno. Si tratta tuttavia di macchinari molto costosi e di difficile reperibilità.

L'errore introdotto nella stima non è tollerabile, pertanto questa configurazione non può essere utilizzata per il test.

Se tuttavia ci si limita ad utilizzare coppie di microfoni collegati alla stessa scheda audio è possibile, utilizzando entrambe le schede, applicare l'algoritmo di stima dell'angolo ai segnali ricevuti da due coppie di microfoni differenti e quindi risalire alla posizione della sorgente

2.4 Microfoni

Per testare al meglio l'algoritmo sono stati selezionati microfoni di qualità medio-alta solitamente utilizzati nell'ambito musicale professionale. I microfoni impiegati sono 4 e sono:

TBone SC 450: Microfono da studio a condensatore a diaframma largo. Ha un range di frequenza dai 20Hz agli 80Hz, un'attenuazione possibile di 10dB ed un filtro passa basso selezionabile fisicamente tramite switch. [11]

Shure PG 81 È un microfono a condensatore cardioide dotato di elevata sensibilità, utilizzato solitamente in studio di registrazione. [12]

Shure SM 58 È il microfono in assoluto più utilizzato in ambito musicale; si tratta di un microfono dinamico con caratteristica polare cardioide. Per il test degli algoritmi sono stati utilizzati due di questi dispositivi. [13]

I microfoni sono stati collegati alle schede audio mediante cavi XLR bilanciati e schermati per ridurre al minimo il rumore.

Figura 2.5: TBone SC 450 [11]

Figura 2.6: Shure SM 58 [12]

Figura 2.7: Shure PG 81 [13]

Capitolo 3

Interfacciamento dei dispositivi

Un'attenta analisi ha portato alla scelta del hardware migliore per effettuare il test; ora è necessario curare con attenzione l'interfacciamento dei dispositivi descritti nel capitolo precedente.

Per comodità si è scelto di utilizzare MATLAB per gestire l'intero progetto. Questo programma consente infatti l'accesso ai dati delle periferiche collegate al pc ed il loro utilizzo all'interno degli algoritmi progettati.

3.1 Schema del progetto

3.1.1 Collegamenti Hardware

L'elaborazione dati viene svolta dal computer su MATLAB; ad esso sono collegate le due schede audio tramite porte USB 2.0. I drivers delle schede sono stati scaricati dal sito del costruttore ed installati sul pc per facilitarne l'interfacciamento con il software.

Ai due dispositivi sono collegati a coppie i quattro microfoni scelti mediante cavi XLR bilanciati e schermati. I microfoni sono infine posti su apposite aste di sostegno e posizionati nella stanza come descritto nel Capitolo 1.

3.1.2 Blocchi logici

Per implementare al meglio l'algoritmo di stima della posizione si è scelto di suddividere il codice in tre funzioni logiche operanti in cascata. Abbiamo quindi:

- Record: funzione che realizza l'interfacciamento tra MATLAB e la scheda audio esterna ed il campionamento del segnale audio ricevuto in ingresso dai microfoni ad essa collegati.

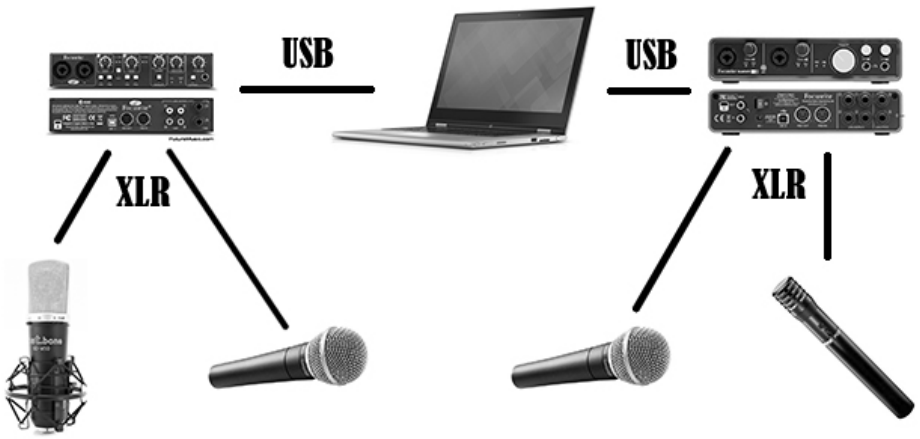

Figura 3.1: Interfacciamento hardware dei dispositivi

- DoA Extimation: funzione che, dopo aver ricevuto in ingresso i vettori contenenti i campioni dei rispettivi segnali audio acquisiti, implementa l'algoritmo di stima dell'angolo di arrivo come descritto nel Capitolo1.
- Source Position: questo blocco logico riceve in ingresso le stime di angolo di arrivo individuate dai campioni acquisiti rispettivamente dalle due coppie di microfoni ed implementa l'algoritmo di triangolazione della posizione della sorgente.

Per facilitare l'utilizzo dell'intero sistema si è scelto di realizzare una *GUI* (*Graphic User Interface)* in grado di gestire e coordinare le tre funzioni di cui sopra.(Capitolo 4)

3.2 Registrazione audio da schede esterne

Come già descritto precedentemente la fase di registrazione è fondamentale per la corretta stima della posizione della sorgente audio. È infatti sufficiente

Figura 3.2: Schema a blocchi logici

un ritardo, anche se minimo, tra due microfoni collegati alla stessa scheda per alterare notevolmente il risultato della stima ed incorrere in una misura errata.

Pertanto è necessario porre la giusta attenzione alle tempistiche di accesso, dovute a MATLAB, alle due schede (che, ricordo, sono interfacciate con il computer mediante protocollo USB 2.0).

La guida MATLAB realizzata da MathWorks fornisce già alcune funzioni in grado di acquisire dati dalle schede audio collegate al computer, indipendentemente dal fatto che esse siano interne od esterne.

3.2.1 Registrazione audio

Per memorizzare dati in ingresso da una periferica audio ed elaborarli successivamente in MATLAB è necessario:

- Creare un oggetto *audiorecorder*.

  ```
  recObj = audiorecorder;
  ```

  L'oggetto in questione è un "registratore" a frequenza di campionamento 8000 Hz, 8 bit per campione ed un solo canale(mono). È necessario per comunicare a MATLAB l'intenzione di acquisire dati dalla periferica audio che il sistema considera come predefinita.

  ```
  Audiorecorder Properties:

  SampleRate: 8000
  BitsPerSample: 8
  NumberOfChannels: 1
  DeviceID: -1
  ```

- Chiamare la funzione *record* o *recordblocking*.

  - *Record:* avvia la registrazione in background permettendo così nel frattempo al software di svolgere eventuali altre istruzioni. Può essere specificata la lunghezza della registrazione in secondi od in alternativa possono essere chiamate le funzioni stop,pause e resume rispettivamente per fermarla, metterla in pausa o riprenderla. La registrazione è gestita in modo asincrono.
  - *RecordBlocking:* avvia la registrazione e blocca lo svolgimento di qualsiasi altra funzione fino al suo termine; anche in questo caso è possibile specificare la durata dell'operazione in secondi. La registrazione è gestita in modalità sincrona.

- Memorizzare i dati in un array numerico mediante la funzione *getaudiodata*.

```
y = getaudiodata(recObj);
```

3.2.2 Gestione delle periferiche esterne

Dopo aver mostrato come sia possibile memorizzare dati in ingresso da una periferica audio è necessario selezionare, utilizzando MATLAB, le schede audio esterne collegate attraverso le porte USB 2.0.

Il software fornisce la funzione *audiodevinfo* che permette di stampare a video la lista delle periferiche audio collegate al pc.

```
info = audiodevinfo;
```

Per esempio questa istruzione fornisce un array contenente la lista di tutte le periferiche audio input/output connesse. Per selezionare una determinata scheda audio è sufficiente specificare il suo ID nella creazione dell'oggetto *audiorecorder*.

```
recorder1 = audiorecorder(44100,16,1,3);
recorder2 = audiorecorder(48000,16,1,4);
```

In questo modo sono stati creati gli oggetti:

- *recorder1*: registratore operante alla frequenza di campionamento 44100 Hz, 16 bit per campione, 1 canale mono e funzionante con la periferica corrispondente al ID 3.

- *recorder2*: registratore operante alla frequenza di campionamento 44800 Hz, 16 bit per campione, 1 canale mono ed associato alla periferica corrispondente al ID 4.

Per registrare da ambedue i dispositivi è sufficiente avviare la funzione record per entrambi gli oggetti creati.

```
record(recorder1);
record(recorder2);
pause(10);
```

La pausa di 10 secondi permette di impostare la durata della registrazione. Per terminarla è necessaria una funzione di stop.

```
stop(recorder1);
stop(recorder2);
```

3.2.3 Blocco logico "Record"

All'interno del blocco logico "Record" sono state utilizzate le funzioni appena descritte in modo da ottenere i campioni dei file registrati dalle due schede audio. La frequenza di campionamento dei due dispositivi è di default pari a 44100 Hz.

```
record_ext1 = audiorecorder(44100,16,2,1);
record_ext2 = audiorecorder(44100,16,2,2);
```

I due registratori così inizializzati permettono l'accesso alle due schede audio. L'unico particolare che li differenzia dagli oggetti descritti finora è che in questo caso si tratta di registratori stereo: l'array ottenuto dalla memorizzazione dei dati in ingresso sarà quindi una matrice a due colonne contenenti rispettivamente i campioni del canale sinistro e destro della scheda.

```
record(record_ext1);
record(record_ext2);
pause(5);
stop(record_ext1);
stop(record_ext2);
```

In questo modo viene avviata la registrazione su entrambi i dispositivi. È fondamentale l'utilizzo della funzione *record* al posto di *recordblocking* in quanto le due azioni devono essere eseguite in contemporanea, nei limiti del possibile; la prima permette infatti di eseguirle in background a differenza della seconda che "congela" il programma fino al termine della registrazione.

Nonostante l'utilizzo di *record* il sistema presenta comunque una latenza ineliminabile tra le due operazioni, dovuta al ritardo generato durante l'accesso alle due schede audio.

La soluzione a questo problema è tuttavia già stata ampiamente discussa nel Capitolo 2.

Successivamente vengono generati gli array contenenti i dati ricevuti dai 4 canali:

```
audio_ext1 = getaudiodata(record_ext1);
audio_ext2 = getaudiodata(record_ext2);
```

Avviene poi la separazione dei rispettivi canali destro e sinistro:

```
L_ext1 = audio_ext1(:,1);
R_ext1 = audio_ext1(:,2);
L_ext2 = audio_ext2(:,1);
R_ext2 = audio_ext2(:,2);
```

Ed infine la memorizzazione dei dati in file audio nel formato Wave mediante la funzione *wavwrite*:

```
wavwrite(L_ext1,fs,16,'outputLX1.wav');
wavwrite(L_ext2,fs,16,'outputLX2.wav');
wavwrite(R_ext1,fs,16,'outputRX1.wav');
wavwrite(R_ext2,fs,16,'outputRX2.wav');
```

Capitolo 4

Graphic User Interface

Ora che sono stati definiti ed interfacciati tutti i componenti necessari al test dell'algoritmo si può procedere con la descrizione della fase di progettazione della *GUI* o *Graphic User Interface* (*Interfaccia grafica per l'utente*).

Le *GUI* semplificano notevolmente la fase di esecuzione del programma permettendone l'utilizzo anche da parte di chi non ha particolari competenze o conoscenze del linguaggio di programmazione utilizzato. Si tratta di applicazioni autonome e personalizzabili che permettono di gestire e richiamare funzioni(anche molto complesse) mediante un semplice click.

In questo caso per realizzare l'interfaccia è stato utilizzato un tool messo a disposizione da MATLAB stesso; il suo nome è *GUIDE* [14].

4.1 GUIDE

Questo ambiente di sviluppo permette di realizzare graficamente la struttura della propria GUI; durante il salvataggio è il software stesso a generare automaticamente lo scheletro del codice necessario al funzionamento dell'interfaccia. Tale codice potrà poi essere modificato per programmare il comportamento dell'applicazione, introducendo al suo interno le funzioni corrispondenti ai blocchi logici descritti nei capitoli precedenti.

Per avviare il tool è sufficiente digitare *guide* nella finestra di comando di MATLAB. All'avvio dell'applicazione è possibile selezionare un nuovo progetto vuoto, uno dei modelli predefiniti od in alternativa aprire un progetto già creato in precedenza.

La finestra di lavoro risulta estremamente semplice ed intuitiva; è possibile visualizzare in tempo reale le modifiche apportate al layout dell'applicazione ed introdurre nuovi elementi a scelta dal menù a sinistra.

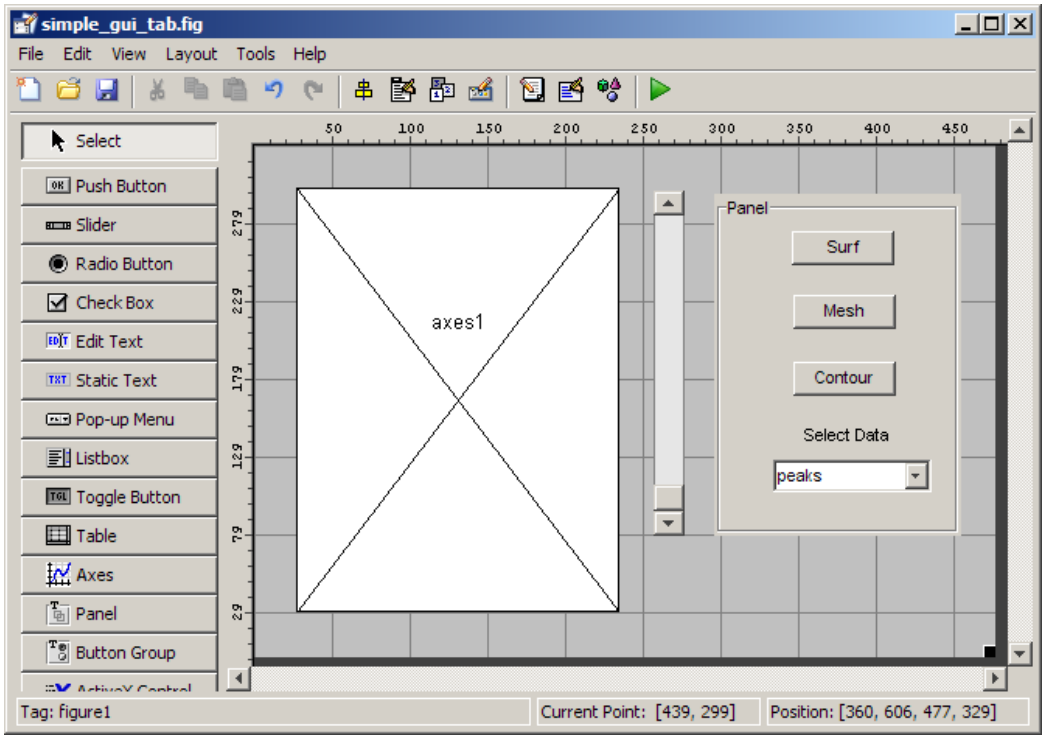

Figura 4.1: Esempio di finestra di lavoro del tool GUIDE.

Ad esempio per inserire un pulsante a pressione all'interno della GUI è sufficiente cliccare sul tasto *Push Button* contenuto nel menù a sinistra e riposizionare l'oggetto che comparirà istantaneamente nell'area di lavoro.

La stessa operazione può essere ripetuta per introdurre un sistema di assi cartesiani, un blocco di testo, etc..

L'applicazione fornisce inoltre uno strumento che permette di allineare i diversi componenti a piacere. Il percorso per accedervi è *Tools - Align Objects*.

Il risultato ottenuto mediante l'utilizzo di GUIDE è un'interfaccia grafica paragonabile a quelle utilizzate quotidianamente ma realizzata senza bisogno di una preparazione specifica sui linguaggi di programmazione comunemente utilizzati.

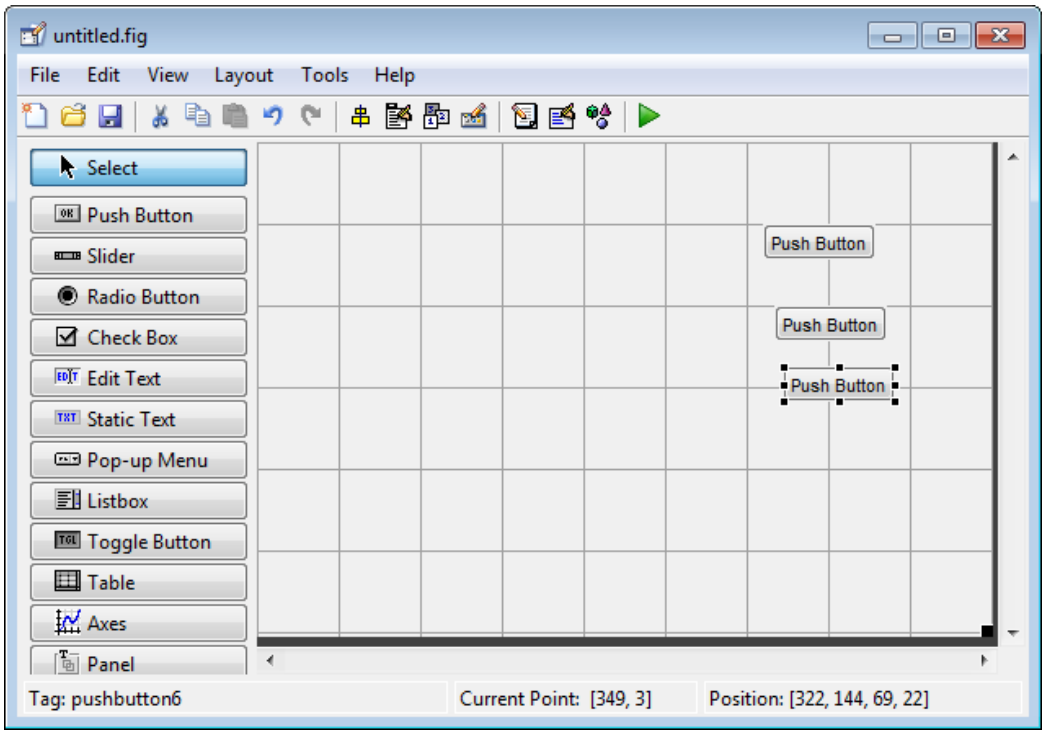

Figura 4.2: Inserimento di *Push Button*.

I singoli oggetti introdotti nello spazio di lavoro possono essere configurati e gestiti effettuando su di essi un doppio clic. In questo modo si accede al menù contenente proprietà e funzionalità di ciascun blocco.

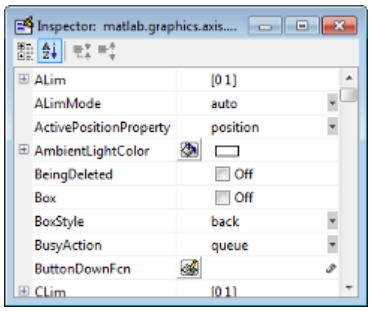

Figura 4.3: Proprietà di un oggetto.

4.2 Realizzazione della GUI

L'interfaccia utente utilizzata per il test dell'algoritmo è suddivisa in quattro sezioni:

- Grafico.
- Menù di regolazione delle distanze tra i microfoni.
- Menù di regolazione della posizione della seconda coppia di microfoni.
- Visualizzazione delle coordinate della posizione della sorgente audio.

Il grafico è realizzato introducendo un sistema di due assi cartesiani e dimensionandoli accuratamente.
I due menù, oltre a presentare la possibilità di modifica di alcune delle grandezze fondamentali per la stima della posizione, contengono due tipi di blocchi:

- *Static Text:* contenente le informazioni sul tipo di dato da inserire nell'altro blocco ed il cui contenuto non è modificabile dall'utente.
- *Edit Text:* visualizzato dall'utente come uno spazio bianco in cui è possibile inserire le informazioni richieste; il contenuto è quindi modificabile dall'utente stesso.

Dopo aver introdotto i dati è necessario cliccare sul tasto *Record*, posizionato in alto sopra al grafico. Il clic avvierà la procedura per il calcolo delle coordinate della posizione della sorgente acustica ed aggiornerà in breve tempo sia il grafico che le coordinate stesse.

4.2.1 Programmazione dell'interfaccia

Dopo aver definito graficamente l'applicazione è necessario aggiungere il codice che permette, alla pressione del tasto *Record*, di avviare l'algoritmo.
Quando viene aggiunto un tasto alla finestra di lavoro il sistema genera automaticamente una funzione pronta ad essere eseguita alla pressione del tasto stesso:

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

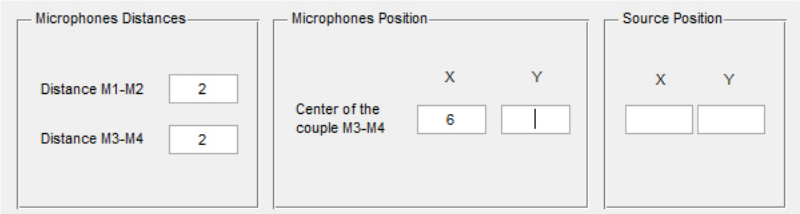

Figura 4.4: Sezione di inserimento dati della GUI.

In questo caso *pushbutton1* è il tag associato al tasto *Record*.

Se all'interno della funzione si utilizza il comando *plot* è necessario comunicare precedentemente al sistema qual'è il grafico su cui si vuole intervenire. Per farlo si utilizza il comando *handles* che permette di selezionare gli elementi presenti nell'area di lavoro.

```
axes(handles.axes1);
```

È inoltre necessario che alla pressione del tasto il sistema legga i valori introdotti dall'utente nei campi dell'interfaccia.
I primi due sono le distanze tra due microfoni appartenenti alla stessa coppia e sono fondamentali per la stima dell'angolo di arrivo.

```
d1 = str2double(get(handles.edit1,'string'));
d2 = str2double(get(handles.edit3,'string'));
```

Le coordinate relative alla posizione della seconda coppia di microfoni sono invece fondamentali nel calcolo della posizione della sorgente audio. È sufficiente una piccola imprecisione nell'inserimento di uno di questi valori per introdurre un errore notevole nel calcolo del risultato finale.

```
xc = str2double(get(handles.edit5,'string'));
yc = str2double(get(handles.edit6,'string'));
```

I dati così acquisiti vengono poi dati in ingresso alla funzione *position* che avvia la registrazione dalle schede audio esterne e l'algoritmo discusso nei capitoli precedenti.

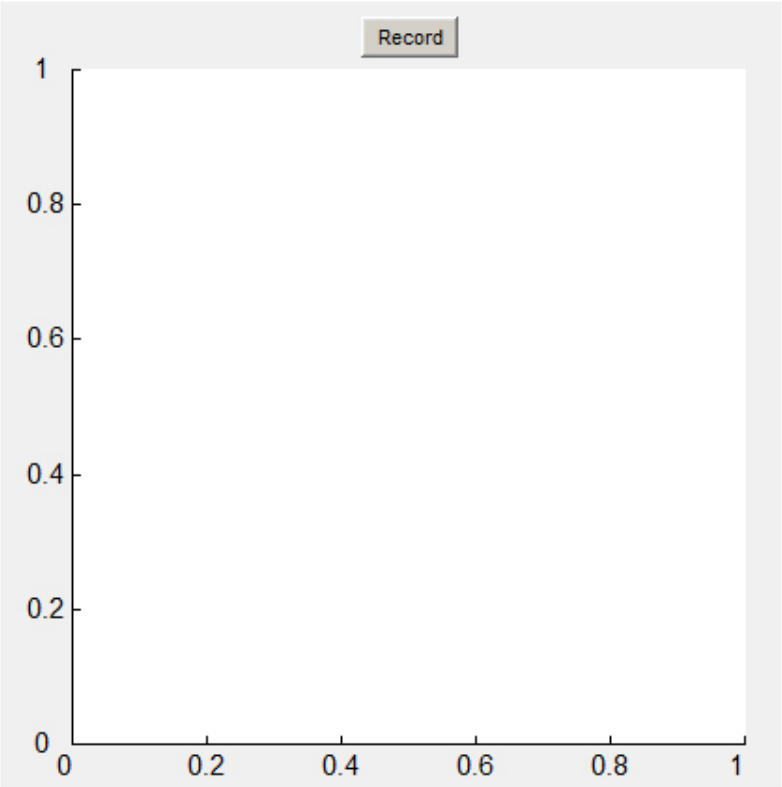

Figura 4.5: Grafico e tasto *Record*.

```
[xp,yp]=position(thetadeg1,180-thetadeg2,d1,d2,xc,yc);
```

Oltre ad aggiornare il grafico dell'applicazione essa fornisce in uscita, dopo averle calcolate, le coordinate della sorgente; queste devono essere infine inserite nei rispettivi blocchi presenti nell'interfaccia.

```
set(handles.edit9,'string',xp);
set(handles.edit10,'string',yp);
```

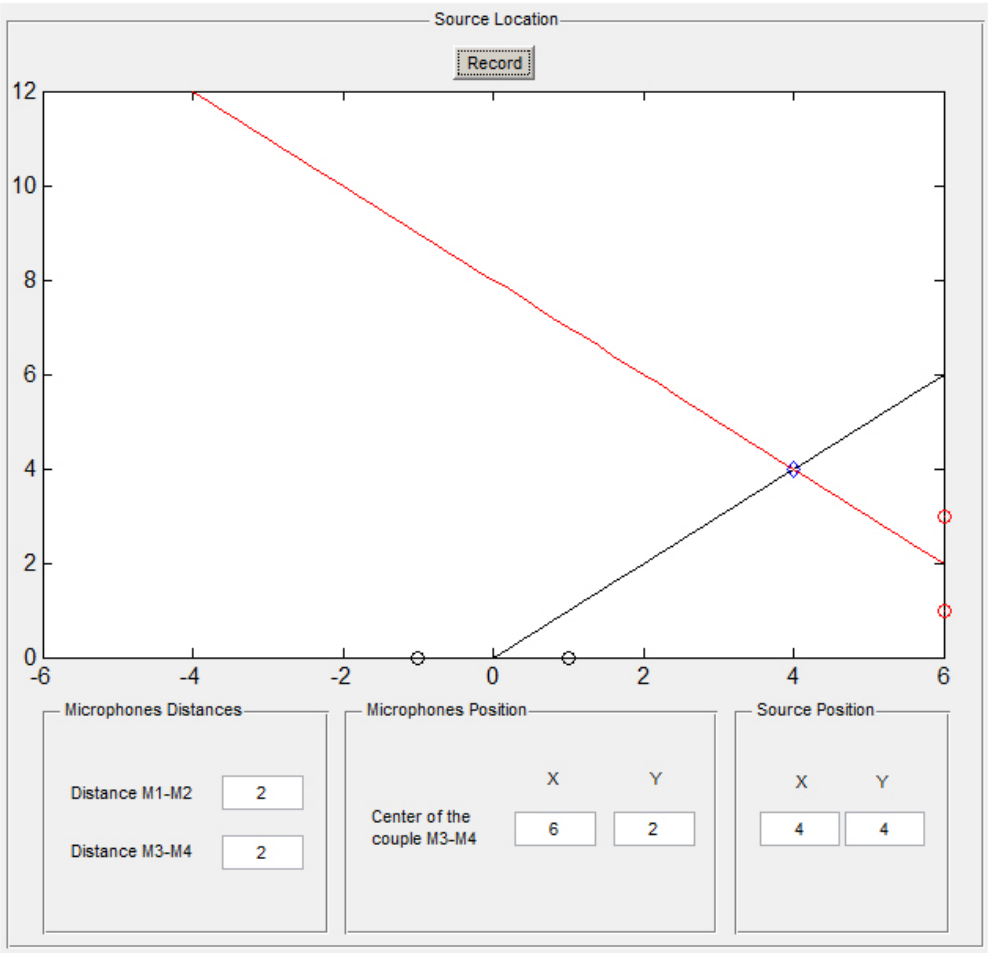

Figura 4.6: Risultato finale.

Capitolo 5

Conclusioni

Dopo numerosi test di funzionamento dell'intero sistema possono essere tratte le dovute conclusioni.

L'algoritmo di stima della direzione di arrivo di un suono e della posizione della sorgente dello stesso si è rivelato particolarmente efficiente. Si è infatti dimostrato funzionante anche in ambienti poco consoni a questo tipo di simulazioni, in presenza cioè di riverberi ed eco che avrebbero potenzialmente potuto introdurre errori nelle misurazioni.

I dispositivi hardware selezionati per il test non hanno causato problemi e si sono rivelati particolarmente efficienti: la qualità delle registrazioni risulta infatti estremamente elevata. Con buona probabilità questo fattore ha influito positivamente sulla precisione dell'algoritmo.

L'interfaccia grafica realizzata risulta particolarmente intuitiva e velocizza notevolmente la fase di simulazione dell'intero processo.

Questo trattato vuole inoltre fungere da guida per chiunque volesse testare in modo rigoroso l'algoritmo ed apportarvi dei miglioramenti. Oltre a quantificarne la precisione, il suddetto potrebbe infatti essere programmato per operare in tempo reale.

Elenco delle figure

Bibliografia

[1] Scott, James, Dragovic, Boris - *Audio location: Accurate Low-Cost Location Sensing*. (Intel research Cambridge 2005)

[2] Parhizkari, Parvaneh: *Binaural Hearing-Human Ability of Sound Source Localization*. Master Thesis in Electrical Engineering Blekinge Institute of Technology.Karlskrona, Sweden. (December 2008)

[3] Calandrino, Chiani: *Lezioni di comunicazioni elettriche*.(2013)

[4] Mathworks - Guida a MATLAB

`http://it.mathworks.com/academia/student_center/tutorials/launchpad.html`

[5] International Civil Aviation Organization

`http://www.icao.int`

[6] Iain McCowan - *Microphone Arrays: A Tutorial*.

`https://www.idiap.ch/~mccowan/arrays/tutorial.pdf`

[7] ETS Solutions

`http://etssolution-asia.com/testing-analysis/measurement-and-analysis/transducers-accessories/microphones/`

[8] Digilent Analog Discovery

`http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,842,1018&Prod=ANALOG-DISCOVERY&CFID=17548965&CFTOKEN=2fee1c74d03c337d-04D02043-5056-0201-023658403CA5B92F`

[9] Focusrite Sapphire Pro 6 USB

http://us.focusrite.com/usb-audio-interfaces/saffire-6-usb

[10] Focusrite Scarlett 2i4 USB

http://us.focusrite.com/usb-audio-interfaces/scarlett-2i4

[11] TBone SC 450

http://www.tbone-mics.com/en/product/information/details/sc450-studio-grossmembranmikrofon/

[12] Shure SM 58

http://www.shure.it/prodotti/microfoni/sm58

[13] Shure PG 81

http://www.shure.com/americas/products/microphones/pg-alta/pga81-cardioid-condenser-instrument-microphone

[14] GUI MATLAB

http://it.mathworks.com/discovery/matlab-gui.html

Ringraziamenti

Desidero ricordare tutti coloro che mi hanno in qualche modo aiutato nella realizzazione di questo elaborato.

Ringrazio anzitutto il professor A. Giorgetti, relatore, per la pazienza e la disponibilità: senza il suo supporto questo lavoro non esisterebbe.

Un ringraziamento particolare va poi ad amici e conoscenti, i quali mi hanno sostenuto ed hanno contribuito con consigli ed osservazioni al miglioramento dell'elaborato.

Vorrei infine ringraziare sentitamente i miei genitori che, con i loro sacrifici, mi hanno permesso di giungere fin qui.