POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Conservatorio
di Milano

Pitch detection attraverso l'auto-correlazione

Elaborato del corso
Teorie dell'ascolto e della percezione sonora e musicale

Autore: **Armando Boemio**

Codice Persona: 1076810
Anno Accademico: 2021-22

Abstract

Tra i metodi più utilizzati nell'ambito di pitch detection, l'analisi dell'autocorrelazione è sicuramente tra i più noti e studiati. Nel seguente elaborato, è descritto il funzionamento generale degli algoritmi di pitch detection che sfruttano l'autocorrelazione, con particolare attenzione a tre diverse implementazioni caratterizzate da complessità e precisione crescente. Nello specifico, sono analizzati segnali vocali (parlato e cantato), che richiedono un'analisi in short-time. Saranno presentati un semplice algoritmo di pitch detection, un algoritmo che prevede una pre-elaborazione del segnale ed infine il noto algoritmo YIN [1]. Delle prime due tipologie è descritta un'implementazione in linguaggio Python. Un capitolo è dedicato alla descrizione del fenomeno del virtual pitch e quindi alla teoria della fondamentale mancante, essendo il pitch tracking con autocorrelazione un metodo estremamente robusto in caso di fondamentale mancante. I risultati prodotti dai diversi algoritmi sono infine confrontati.

Parole chiave: pitch, pitch detection, correlazione, autocorrelazione, virtual pitch

Indice

Introduzione

La frequenza fondamentale F_0 di un segnale periodico è definita come l'inverso del suo periodo, ovvero il minimo tempo dopo il quale il segnale si ripete uguale a sè stesso. Questa definizione è valida esclusivamente per segnali perfettamente periodici, che tuttavia sono anche poco interessanti dal punto di vista uditivo e musicale, dato che ogni modulazione causerebbe la perdita di tale periodicità perfetta. I segnali musicali e del parlato si distaccano di molto da questa definizione di periodicità, pertanto trovarne la frequenza fondamentale è un operazione non sempre semplice che va affrontata con metodi solidi e robusti.

Il pitch è una grandezza soggettiva psicoacustica, che indica quanto acuto o grave viene percepito un suono. La frequenza del pitch è generalmente collegata alla frequenza fondamentale e spesso corrisponde strettamente ad essa, ovvero corrisponde alla frequenza di ripetizione del segnale. Storicamente, sono state date diverse definizioni di pitch, anche a seconda dell'ambito in cui esso è utilizzato. Ad esempio, un suono può essere aperiodico e creare comunque la sensazione di un pitch (Miller and Taylor, 1948)[5]. Un'altra definizione di pitch è la "risoluzione del pattern armonico delle frequenze parziali, risolte dalla coclea nel dominio della frequenza" (Goldstein, 1973) [2]. Infine, può essere utilizzato indistintamente come F_0, come nel caso degli algoritmi di pitch detection che sono affrontati nell'elaborato.

Per i segnali vocali, la frequenza fondamentale è definita come la frequenza di vibrazione delle corde vocali. Il parlato tuttavia risulta essere molto meno periodico a causa del filtraggio che avviene attraverso il tratto vocale. Inoltre la vibrazione stessa può presentare aperiodicità e cambi di ampiezza e forma d'onda, a seconda della tecnica utilizzata per produrre un determinato suono. Tutto questo rende molto complesso il compito di stimare la frequenza fondamentale di un suono parlato o cantato.

Nel corso egli anni sono stati studiati e proposti diversi metodi, quali AMDF (Average Magnitude Difference Function), HPS (Harmonic Product Spectrum), PV (Phase Vocoder), CV (Channel Vocoder) o metodi più moderni basati sulla frequenza istantanea o su reti neurali. Rimangono rilevanti, infine, gli approcci basati sull'autocorrelazione o sue

dirette evoluzioni, anche grazie alla precisione di tali algoritmi rispetto al fenomeno della fondamentale mancante, approfondito al capitolo 3.

Nel seguente elaborato sono presentate le basi dei metodi che utilizzano l'autocorrelazione e tre diversi approcci di implementazione. Di questi, due sono stati implementati ex-novo in ambiente Python, mentre il terzo è stato testato sulla base del codice originale fornito degli autori. Il terzo capitolo è dedicato alla teoria del virtual pitch, mentre il capitolo finale presenta un confronto tra le prestazioni dei diversi algoritmi.

1 | Analisi dell'autocorrelazione

1.1. Funzione di autocorrelazione

La funzione di autocorrelazione (ACF) di un segnale è definita come la correlazione di un segnale con una copia ritardata del segnale stesso. Descrive quanto un segnale è simile a se stesso in funzione di un parametro temporale detto "lag".

Essa è estremamente utile quando è necessario cercare un pattern periodico in una funzione rumorosa, o quando è necessario trovare la fondamentale mancante in una serie armonica.

Nel campo dell'elaborazione dei segnali si ha spesso a che fare con segnali deterministici sia in tempo continuo che discreto.

1.1.1. Autocorrelazione di segnali in tempo continuo

Dato un segnale tempo continuo $f(t)$, l'autocorrelazione $R_{ff}(\tau)$ è definita come:

$$R_{ff}(\tau) = \int_{-\infty}^{+\infty} f(t)\overline{f(t-\tau)}dt \tag{1.1}$$

dove τ è il fattore di lag (misurato in secondi) e $\overline{f(t)}$ rappresenta il complesso coniugato.

1.1.2. Autocorrelazione di segnali in tempo discreto

Dato un segnale tempo discreto $x(n)$, l'autocorrelazione $R_{xx}(m)$ è definita come:

$$R_{xx}(m) = \sum_{n\in\mathcal{Z}} x(n)\overline{x(n-m)} \tag{1.2}$$

dove m è il fattore di lag (misurato in samples) e $\overline{x(n)}$ rappresenta il complesso coniugato.

1.1.3. Autocorrelazione di segnali continui periodici

Nel caso di segnali continui periodici di periodo T, l'integrazione da $-\infty$ a $+\infty$ è sostitutita da un integrazione nell'intervallo $[t_0, t_0 + T]$ di lunghezza T:

$$R_{ff}(\tau) \triangleq \int_{t_0}^{t_0+T} f(t)\overline{f(t-\tau)}dt \tag{1.3}$$

1.2. Proprietà dell'autocorrelazione

L'autocorrelazione gode di diverse interessanti proprietà, valide nel caso mono dimensionale per processi stazionari in senso lato:

- Simmetria: la funzione di autocorrelazione è una funzione pari quando il segnale f è reale, ovvero:

$$R_{ff}(\tau) = R_{ff}(-\tau). \tag{1.4}$$

- periodicità: l'autocorrelazione di una funziona periodica di periodo T è essa stessa periodica dello stesso periodo:

$$R_{ff}(\tau) = R_{ff}(\tau + T). \tag{1.5}$$

- Picco nell'origine: l'autocorrelazione ha un picco nell'origine, dove il suo valore è reale. Pertanto, per qualsiasi lag τ, vale

$$|R_{ff}(\tau)| \leq R_{ff}(0). \tag{1.6}$$

  Questo risultato è valido sia in tempo continuo che in tempo discreto ed è di fondamentale importanza nell'implementazione degli algoritmi di pitch detection.

- Convoluzione: l'autocorrelazione può essere anche scritta in termini di convoluzione. Considerando una funzione $g_{-1}(f)(t) = f(-t)$, è possibile scrivere

$$R_{ff}(\tau) = \left(f * g_{-1}\left(\overline{f}\right)\right)(\tau). \tag{1.7}$$

1.3. Analisi in short-time

Ai fini del rilevamento del pitch, se si considerassero segnali f o x perfettamente periodici, le definizioni appena presentate sarebbero sufficienti per un'analisi completa. Tuttavia, tutti i segnali di interesse (suoni, voce parlata, voce cantata) sono segnali non stazionari, ovvero cambiano nel tempo in ampiezza e frequenza, pertanto un'analisi diretta di un segnale canoro utilizzando l'autocorrelazione non produrrebbe alcun risultato significativo. Per questo motivo è necessario operare in "short-time", ovvero segmentando il segnale in brevi finestre temporali ed analizzandone il contenuto finestra per finestra.

Si definisce quindi autocorrelazione in short time:

$$\phi_l(m) = \frac{1}{N} \sum_{n=0}^{N'-1} [x(n+l)\, w(n)]\, [x(n+l+m)\, w(n+m)] \quad 0 \leq m \leq M_0 - 1 \tag{1.8}$$

dove

- $w(n)$ è la finestra di analisi;
- N è la lunghezza della sezione che si sta analizzando;
- N' è il numero di samples del segnale che sono utilizzati nel calcolo dell'autocorrelazione;
- M_0 è il numero di punti dell'autocorrelazione processati;
- l è l'indice del punto iniziale del frame considerato.

In generale, per le applicazione di pitch detection si definisce

$$N' = N - m \tag{1.9}$$

in modo che solo N campioni del frame di analisi sono usati nel calcolo dell'autocorrelazione.

I valori delle lunghezze N ed M_0 sono variabili e dipendono dall'applicazione. Allo stesso modo, il tipo di finestra utilizzata influisce sulle prestazioni del modello. Tutte le problematiche relative all'implementazione sono discusse nel capitolo successivo.

2 | Algoritmi di Pitch Detection

Un algoritmo di pitch detection (PDA - Pitch Detection Algorithm), è un algoritmo il cui obiettivo è stimare il pitch o la frequenza fondamentale di un segnale quasi-periodico o oscillante. Generalmente si applica a segnali vocali o musicali. Al giorno d'oggi non esiste un'unica soluzione ideale adatta ad ogni problema, ma ogni sistema è più adatto a risolvere uno specifico tipo di problematica. Gli algoritmi stimano il periodo del segnale in analisi e la frequenza (o pitch) è ottenuta quindi per inversione.

Nel seguente capitolo sono analizzati i metodi che si basano sull'utilizzo della autocorrelazione.

2.1. Metodo dell'autocorrelazione

Come esposto in sezione 1.2, tra le proprietà fondamentali della funzione di autocorrelazione figurano la periodicità ed il picco nell'origine. In particolare, quindi, con l'aumentare del valore del lag τ, la correlazione raggiunge un valore di minimo, per poi crescere nuovamente quando il valore del lag si avvicina al valore del periodo. Questo comportamento è dovuto al fatto che le forme d'onda ritardate sono dapprima in fase, poi perfettamente fuori fase in corrispondenza di metà periodo, e di nuovo in fase in corrispondenza di un intero periodo.

Tuttavia, questo è vero soltanto per segnali semplici. Per segnali armonicamente complessi bisogna considerare una forma d'onda pseudo-periodica. Il primo picco in questo caso non è rilevato in corrispondenza del periodo dell'intera forma d'onda, ma al periodo corrispondente ad una n-esima armonica superiore. Si tratta tuttavia di un picco "minore", in quanto il primo picco "maggiore" si mostrerebbe, in ogni caso, in corrispondenza della frequenza fondamentale dell'onda. Una delle maggiori sfide nella progettazione di algoritmi di pitch detection basati sull'autocorrelazione è proprio la capacità di distinguere picchi "maggiori" e "minori" consistentemente, in quanto è facile sbagliare la scelta del picco ed ottenere quindi una sovrastima, o sottostima, del pitch.

Fortunatamente, la capacità computazionale non è l'unico elemento d'aiuto nella scelta

corretta del pitch. È innanzitutto noto che per un segnale vocale, il range di frequenza del pitch è ben noto e limitato. Statisticamente, il pitch rientra in un intervallo che va dai 40 Hz per voci "low-pitched", generalmente maschili, fino ai 600 Hz per voci "high-pitched", generalmente femminili o di bambini. In particolare per i suoni nei linguaggi latini, a seconda del modo in cui esso è generato si distinguono tre classi: suoni vocali, suoni fricativi e suoni occlusivi. Inoltre, è dimostrato che i suoni fricativi e occlusivi non hanno pitch [3].

I metodi con autocorrelazione richiedono almeno due periodi di forma d'onda per rilevare un pitch. Questo implica che più è basso il pitch da trovare, più lunga deve essere la finestra di analisi. Tuttavia non è detto che in un intervallo di tempo abbastanza lungo il pitch rimanga costante, essendo la voce un segnale tendenzialmente non stazionario. Un'analisi più approfondita di questa problematica è affrontata nella sezione 2.1.3.

Nelle sezioni successive sono descritte due possibili tra le possibili implementazioni del metodo dell'autocorrelazione. I codici sono disponibili nella repository di Github al seguente link: `https://github.com/armandoboemio98/PitchDetector`.

2.1.1. Implementazione triviale

Il primo algoritmo, implementato nel file `simple_acf.py` della repository, è un semplice pitch detector che calcola l'autocorrelazione su tutto il segnale in ingresso. L'algoritmo si sviluppa come segue:

Algorithm 2.1 simple_acf

1: Load audio data
2: compute_acf
3: find_peaks
4: lag = first peak found
5: compute pitch from lag
6: plot power spectrum and spectrogram

L'algoritmo 2.1 risulta essere l'implementazione più semplice e diretta tra i metodi che utilizzano l'autocorrelazione. Non c'è alcun tipo di preprocessing o finestrazione del segnale. Lo script calcola il pitch per inversione e produce i grafici dello spettrogramma e dello spettro di potenza. Quest'ultima operazione non è necessaria ai fini del pitch tracking ma è utile per verificare il funzionameno dello script stesso.

Come detto in precedenza, un'analisi in short-time è necessaria per la non stazionarietà dei suoni di interesse. Pertanto questo algoritmo non può rilevare correttamente il pitch di un qualsiasi segnale musicale o vocale, ma è in grado di riconoscere il pitch di un segnale pseudo-stazionario, quale un diapason o un semplice oscillatore, con sufficiente precisione, come mostrato in nella sezione 4, dove sono esposti i risultati.

2.1.2. Implementazione short-time con preprocessing

Il secondo algoritmo, implementato nel file `short_time_acf` ha un architettura più complessa rispetto all'algoritmo appena presentato. Questo design è basato sull'archiettura presentata da Rabiner nel 1977 in *On the use of autocorrelation analysis for pitch detection* [7].

Innanzitutto, il segnale viene pre-processato. L'obiettivo di questa operazione è di ridurre l'effetto delle formanti sulle code dell'autocorrelazione ed inoltre rendere il segnale più piatto (spectral flattening). In particolare, è prima filtrato con un filtro digitale di Butterworth passa-basso del sesto ordine. La frequenza di taglio è settata tra i 600 ed i 900 Hz, in modo da preservare abbastanza armoniche per avere un corretto tracciamento del pitch ma eliminare le formanti secondarie. Successivamente, il segnale viene trasformato utilizzando due tipi di non linearità, in modo da rendere lo spettro più piatto. Le funzioni non lineari utilizzate non sono necessariamente uguali. Nel paper di riferimento sono analizzate e confrontate diverse tipologie, mentre nell'implementazione presentata sono scelte solo quelle che riscontrano risultati migliori. Sono utilizzate, quindi, una funzione di clip centrale (clp) ed una funzione di "clip and compress" (clc). Esse sono definite come

$$\texttt{clc}[x(n)] = \begin{cases} (x(n) - C_L), & x(n) \geq C_L \\ 0, & |x(n)| < C_L \\ (x(n) + C_L), & x(n) \leq -C_L \end{cases} \qquad \texttt{clp}[x(n)] = \begin{cases} x(n), & x(n) \geq C_L \\ 0, & |x(n)| < C_L \\ -x(n), & x(n) \leq -C_L \end{cases}$$

dove C_L è la soglia di clipping.

La scelta dell'utilizzo delle nonlinearità è dovuta a due caratteristiche principali. Sul lato computazionale, la loro implementazione è estremamente efficiente, e di fatto l'utilizzo di due diverse non linearità aggiunge flessibilità al sistema. Inoltre, è stato provato che l'utilizzo della correlazione tra le due trasformazioni è soluzione la più appropriata in molte situazioni reali di pitch detection. Pertanto i due output sono dati in ingresso ad un correlatore short-time. In figura 2.1 è mostrato un diagramma a blocchi della struttura del pre-processore.

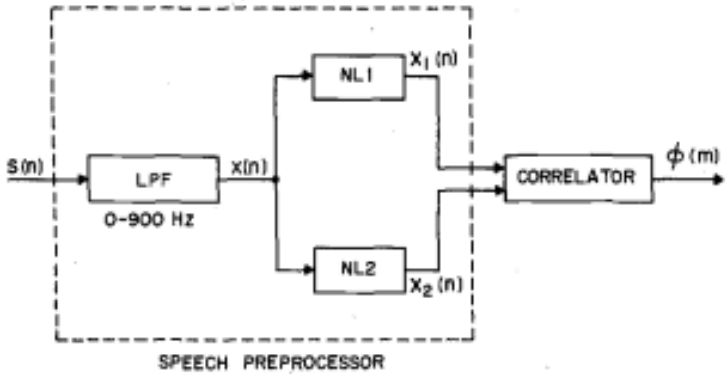

Figura 2.1: Schema originale presentato da Rabiner in [7].

L'algoritmo può essere schematizzato con il seguente pseudo-codice:

Algorithm 2.2 short_time_acf

```
Load audio data
trim_audio
Apply low pass filter
Compute clipping threshold
Apply non linearities
for i in range of #frames do
    correlate processed outputs
    compute lags in range [acf_min:acf_max]
    lag = argmax[x_corr]
    compute pitch from lag
end for
Plot and compare results with YIN
```

Rispetto allo schema originale è stata aggiunta una funzione di trimming automatico dei silenzi all'inizio ed alla fine degli audio. Questo accorgimento serve per ridurre al minimo gli artefatti nella computazione in short time dei frame iniziali e finali dell'audio caricato.

Ai fini del confronto, lo script produce due grafici. Il primo è il valore del pitch in ogni finestra ottenuto utilizzando l'algoritmo 2.2. Il secondo grafico è il valore del pitch ottenuto utilizzando un pitch detector della libreria di audio processing `librosa`, basata sullo YIN, metodo spiegato nella sezione 2.2.

2.1.3. Scelta dei parametri

Un aspetto determinante per il corretto funzionamento dell'algoritmo presentato in 2.1.2 è la scelta dei parametri utilizzati. Tra i più rilevanti si possono indicare:

- **Lunghezza della finestra di analisi**

  Nell'implementazione corrente, la finestra di analisi è una finestra rettangolare lunga 30ms. In letteratura, 30ms è considerato come un valore standard ideale ([3],[7]), sebbene possa essere troppo breve per suoni con pitch estremamente basso. Non è troppo rilevante la tipologia di finestra usata, in quanto il finestramento stesso genera un assottigliamento ("tapering") della funzione di autocorrelazione a 0 con l'aumentare dell'indice di autocorrelazione, a prescindere dal tipo di finestra. Pertanto, per semplicità di implementazione, è stata scelta una finestra rettangolare.

- **Tipologia e frequenza di taglio del filtro passa basso**

  Per la frequenza di taglio, come detto in precedenza, è conveniente settare un valore tra i 600 ed i 900 Hz. In particolare, nello script è stato scelto il valore minimo, 600 Hz, per evitare possibili componenti di rumore oltre il massimo pitch statistico. L'ordine e la tipologia del filtro non sono parametri fondamentali, a patto che il filtro sia il piu trasparente possibile in banda passante, da cui la scelta di un filtro Butterworth massimamente piatto. La scelta di un filtro così semplice ed efficace è anche mirata a migliorare le prestazioni computazionali dell'algoritmo.

- **Soglia delle funzioni di clipping**

  La soglia di clipping è un parametro piuttosto delicato. Non c'è un valore ideale che garantisce il massimo delle prestazioni per ogni sistema o situazione. Pertanto, è necessario affidarsi solo ai test effettuati da altri ricercatori. L'approccio utilizzato in 2.1.2 è il medesimo proposto da Rabiner [7]. Il valore di soglia di clipping è quindi scelto come il 68% del valore minimo trovato tra i massimi delle primo e ultimo terzo delle finestre in cui è diviso il segnale di ingresso. In formula:

$$C_{th} = 0.68 \times min\left[max\left(W_0 : W_m\right), max\left(W_{N-m} : W_N\right)\right] \tag{2.1}$$

  dove W_n è una finestra, N è il numero totale di finestre ed $m = \frac{N}{3}$.

- **Intervallo di frequenze di interesse**

  Il parametro probabilmente più rilevante è l'intervallo in cui limitare la ricerca dei picchi di autocorrelazione. Il motivo è che nel processing di segnali digitali, la ricerca dei picchi può essere non perfettamente precisa, a causa della limitata risoluzione e potenza computazionale. Inoltre possono essere presenti artefatti nel segnale, dovuti a problemi di quantizzazione, finestrazione o burst di rumore nel segnale in ingresso. Pertanto è necessario fornire un intervallo di valori entro cui ricercare picchi o massimi, in modo da minimizzare l'errore prodotto. In particolare, nello script presentato la ricerca è limitata a [`acf_min:acf_max`], ovvero ai lag corrispondenti alle frequenze di interesse del parlato o cantato. I valori sono ottenuti attraverso le seguenti formule, presentate anche in [3]:

$$\begin{cases} \texttt{acf\_min} = \dfrac{F_s}{F_{max}} \\ \texttt{acf\_max} = \dfrac{F_s}{F_{min}} \end{cases} \tag{2.2}$$

  dove

  - F_s è la frequenza di campionamento;
  - F_{min} è la minima frequenza di interesse, settata a 40 Hz;
  - F_{max} è la massima frequenza di interesse, settata a 600 Hz.

2.2. Metodo YIN

Il metodo YIN, proposto da *A. De Cheveigne* e *H. Kawahara* [1], è un metodo basato sull'autocorrelazione e sulla cancellazione. Ad oggi risulta essere tra i più affidabili algoritmi di pitch detection, motivo per cui è utilizzato come termine di paragone e testbench per l'algoritmo 2.2.

Esso nasce come superamento dei tradizionali metodi che utilizzano autocorrelazione, in quanto non sempre affidabile per applicazioni in tempo reale, o in situazioni dove c'è un'alta variazione nel pitch del parlato ed è richiesta un'alta precisione.

Una descrizione approfondita del metodo risulterebbe estremamente lunga e complessa ed è pertanto fuori dagli scopi di questo elaborato. Tuttavia il metodo può essere descritto sufficientemente considerandone diversi step, ognuno dei quali introduce un miglioramento rispetto all'errore nel tracking:

1. **Metodo dell'autocorrelazione**

   Il primo step consiste semplicemente nell'utilizzo dei metodi appena presentati.

2. **Funzione alle differenze**

   L'idea principale è di trovare un periodo non noto di una funzione utilizzando la funzione alle differenze

$$d_t(\tau) = \sum_{j=i}^{W} (x_j - x_{j+\tau})^2 \tag{2.3}$$

   ovvero cercando valori di τ tali per cui la funzione è zero. Esistono infinite soluzioni, corrispondenti ai multipli del periodo. L'equazione (2.3) può essere scritta in funzione della funzione di autocorrelazione una volta estesa:

$$d_t(\tau) = r_t(0) + r_{t+\tau}(0) - 2r_t(\tau) \tag{2.4}$$

   dove r_τ è la funzione di autocorrelazione. I primi due termini sono termini energetici. Se fossero entrambi costanti, non ci sarebbe differenza tra autocorrelazione e funzione alle differenze. Tuttavia, il primo è costante, mentre il secondo varia con τ. Pertanto, massimi e minimi di r_τ e d_τ possono non coincidere.

   Utilizzare la funzione alle differenze comporta un errore nella rilevazione circa cinque volte minore (10% contro 1.95%). Il principale motivo di questa grande differenza è la sensibilità dell'autocorrelazione rispetto ai cambiamenti di ampiezza nel segnale di input. Un aumento di ampiezza nel tempo causano una crescita anche nei lag, che invece dovrebbero rimanere costanti per un corretto rilevamento. Questo comportamento induce gli algoritmi a scegliere picchi di indici superiori, e quindi a sottostimare il pitch. Un comportamento opposto accade per riduzioni dell'ampiezza dell'input, causando una sovrastima del pitch. La funzione alle differenze è invece immune a questo tipo di problema.

3. **Funzione cumulativa alle differenze media normalizzata**

   Spesso, a causa della non perfetta periodicità, la funzione alle differenze è nulla all'indice zero e non nulla all'indice corrispondente al periodo. Pertanto, un algoritmo scritto per trovare gli zeri è destinato a fallire se non viene definito un limite inferiore per la ricerca. Inoltre le formanti secondarie potrebbero creare altre valli che posso-

no essere identificate come possibili pitch. Per risolvere questo problema, è possibile utilizzare la funzione cumulativa alle differenze media normalizzata, definita come

$$d'_t(\tau) = \frac{d_t(\tau)}{\left(\frac{1}{\tau}\right)\sum_{j=1}^{\tau} d_t(j)} \tag{2.5}$$

con $d'_t(\tau) = 1$ per $\tau = 0$.

L'utilizzo di questa funzione ha un triplice vantaggio: riduce gli errori di sovrastima, rimuove la necessità di utilizzare un limite superiore di ricerca e normalizza la funzione. L'errore complessivo è cosi ridotto al 1.69%.

4. **Soglia assoluta**

   Non è raro che le valli di ordine superiore siano piu profonde delle valli corrispondenti al reale periodo. Quando questo accade, c'è un errore di sottostima, ovvero la rilevazione di un ottava inferiore rispetto al pitch, come accade anche nel metodo dell'autocorrelazione diretto. Una possibile soluzione è definire una soglia assoluta di profondità, e scegliere, tra i candidati, il valore di lag minore. L'errore è ridotto fino allo 0.78%, a fronte di un miglioramento nel problema della sottostima ma un lieve peggioramento per quanto riguarda la sovrastima, dato che potrebbe essere scelto un lag troppo piccolo se la soglia non è settata con attenzione.

5. **Interpolazione parabolica**

   L'utilizzo dell'interpolazione parabolica non comporta un miglioramento significativo in condizioni normali, tuttavia diventa rilevante quando si utilizzano passi di campionamento molto larghi, e quindi frequenze di campionamento che possono essere vicine al valore del pitch. In questo caso, la rilevazione può distaccarsi per un valore fino a metà della frequenza di campionamento rispetto al valore target. L'interpolazione consiste nel costruire parabole in corrispondenza dei minimi locali, utilizzando i campioni adiacenti ed è estremamente meno costosa di un eventuale up-sampling del segnale stesso.

6. **Miglior valore locale stimato**

   L'ultimo step risolve infine il problema delle fluttuazioni di pitch su scala temporale del periodo stesso. Questo avviene applicando l'algoritmo una seconda volta per ogni indice temporale t ma in un range minore. Una volta trovato il primo valore, si ricerca un minimo di $d'_\theta(T_\theta)$ dove T_θ è il valore stimato al tempo θ. Il piccolo intervallo di ricerca è definito come $\left[t - \frac{T_{max}}{2}, t + \frac{T_{max}}{2}\right]$, con T_{max} che corrisponde

al massimo periodo previsto. Con questo accorgimento, l'errore si riduce fino allo 0.5%.

Il metodo YIN, quindi, non è altro che la combinazione dei sei step presentati, che si costruiscono l'uno a partire dall'altro. In tabella 2.1 è riassunto il valore degli errori ad ogni step.

Version	Error (%)
Step 1	10.0
Step 2	1.95
Step 3	1.69
Step 4	0.78
Step 5	0.77
Step 6	0.50

Tabella 2.1: Tabella riassuntiva degli errori degli step del metodo YIN.

3 | Virtual Pitch

Per virtual pitch (pitch virtuale) si intende il pitch di un tono complesso, ovvero un suono che può essere scomposto in parziali. Esso corrisponde in genere alla frequenza fondamentale di una serie armonica. La peculiarità del virtual pitch è che può essere tracciato e percepito anche quando la fondamentale è mancante o non perfettamente accordata con la serie armonica.

Viene definito pitch virtuale in quanto non ha un corrispettivo acustico nella frequenza alla quale viene percepito. Di fatto, quando la frequenza fondamentale non è fisicamente presente in un tono o segnale, l'apparato uditivo ed il cervello sono in grado di ricostruirla e riconoscerla a partire dalla serie armonica.

Questo fenomeno rientra nella categoria del riconoscimento dei pattern sul piano psicoacustico. All'inizio degli anni '70, *Ernst Terhardt ed altri* sostennero alcuni esperimenti psicoacustici [9] per meglio chiarire l'idea di pitch. Essi ipotizzarono che la percezione del pitch avveniva in due passaggi: prima un'analisi uditiva della frequenza nell'orecchio interno, e successivamente il riconoscimento di pattern armonici nel cervello. Fu inoltre possibile definire quali parziali fossero percettivamente rilevanti in un dato suono complesso. Questo dipende da aspetti sia temporali che spettrali, in quanto, per essere definite rilevanti, le armoniche dovevano far percepire un suono o pitch diverso quando la loro ampiezza o frequenza era modificata.

Secondo Terhardt [8], c'è inoltre una differenza tra virtual pitch e spectral pitch: il primo è caratterizzato dalla presenza di armoniche, mentre il secondo è caratterizzato da singoli toni puri udibili. Definisce inoltre il passaggio da virtual a spectral pitch intorno agli 800 Hz, sostenendo che per frequenze ben oltre i 1000 Hz, un pitch viene percepito se e solo se la fondamentale è fisicamente presente nel segnale. Si deduce quindi che la maggioranza dei pitch percepiti da suoni comuni sono di fatto virtuali, indipendentemente quindi dalla presenza della fondamentale.

3.1. Teoria della fondamentale mancante

Un suono si dice avere una fondamentale mancante, soppressa o fantasma quando i suoi overtoni suggeriscono una frequenza fondamentale ma il suono è privo di tale componente [10]. Questo è una diretta conseguenza del virtual pitch, dato che il cervello percepisce il pitch di tono non solo grazie alla fondamentale ma anche grazie alla relazione di periodicità delle sue armoniche. Si tratta quindi di un fenomeno psicoacustico, il cui preciso meccanismo è ancora argomento di dibattito. Tra le possibili risposte, è stato speculato che il meccanismo potrebbe essere non dissimile dal un processo di autocorrelazione, che coinvolgerebbe quindi una misurazione del tempo degli impulsi neurali nel nervo uditivo.

In figura 3.1 è possibile visualizzare graficamente il fenomeno della fondamentale mancante. Nel grafico supeirore è raffigurata una serie armonica con fondamentale a 100 Hz. Nel grafico sottostante è rappresentata la stessa serie armonica senza tuttavia la componente fondamentale e la prima armonica. Nonostante le diverse forme d'onda, è chiaro verificare comunque la presenza di una componente periodica a 100 Hz.

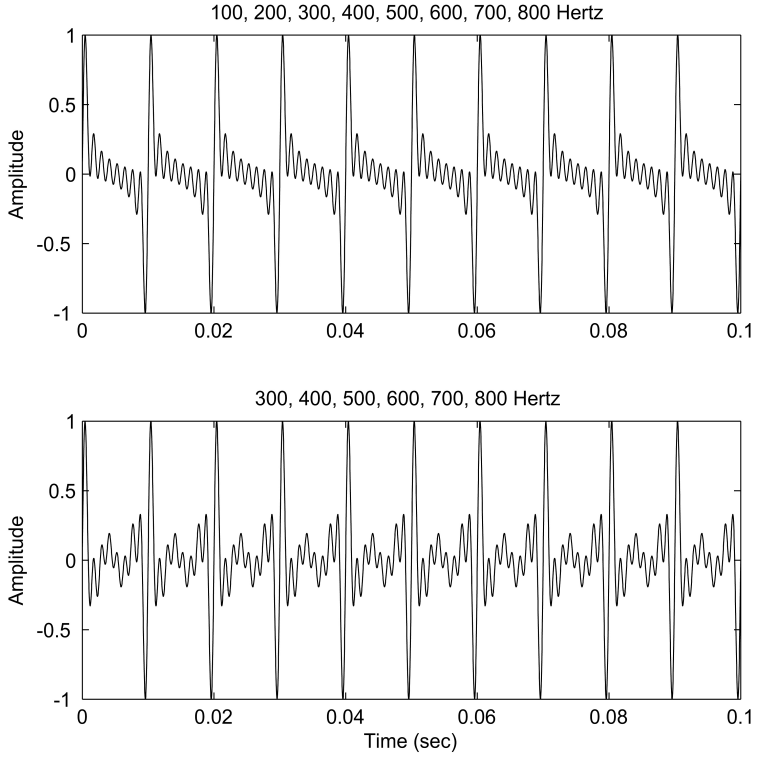

Figura 3.1: Sopra: forma d'onda di una serie armonica completa con fondamentale a 100 Hz. Sotto: forma d'onda di una serie armonica con frequenza fondamentale a 100 Hz senza la fondamentale e la prima armonica.

L'immagine appena presentata suggerisce come l'autocorrelazione risulti essere un metodo estremamente robusto per il rilevamento del pitch virtuale, basando il suo funzionamento sulla ricerca di un periodo comune (spesso il massimo comune divisore) alle armoniche più rilevanti di un segnale acustico.

3.1.1. Utilizzo nel processing audio

La teoria della fondamentale mancante ad oggi è ancora oggetto di continua ricerca. I suoi utilizzi sono numerosi ed in continua crescita, merito soprattutto del processo di miniaturizzazione che sta avvenendo in ogni campo dell'elettronica di consumo. Un esempio proveniente dall'uso quotidiano viene dall'utilizzo di piccole casse, come quelle di smarthphone, laptop, speaker portatili o cuffie in-ear. Molto spesso tali dispositivi sono realizzati usando trasduttori piezoelettrici, la cui risposta in frequenza è approssimativamente costante sopra la frequenza di risonanza f_c, mentre è proporzionale al quadrato della frequenza al di sotto. Solitamente, il valore di f_c si aggira intorno ad 1 Khz, rendendo quindi la riproduzione delle basse frequenze molto scadente ed inefficiente. Una soluzione è pertanto l'utilizzo di algoritmi di amplificazione virtuale dei bassi (VBE - Virtual Bass Enhancement), il cui obiettivo è di allargare psicoacusticamente la banda di basse frequenze, spostando l'energia da una porzione bassa dello spettro ad una più alta (*Larsen and Aarts* [4]).

Ad oggi, esistono diverse tipologie di algoritmi di VBE, ma hanno tutti un paradigma in comune: il segnale viene filtrato in passa-alto, in modo da eliminare fisicamente dallo spettro tutte le componenti che gli speaker non sarebbero in grado di riprodurre, ottimizzando peraltro l'utilizzo energetico, e successivamente viene arricchito spettralmente con la generazione di armoniche sintetiche.

Tra le applicazioni più note rientra sicuramente lo storico plug-in "MaxxBass", realizzato da Waves Audio, in cui la generazione di armoniche è ottenuta utilizzando componenti non lineari, che distorcendo il segnale creano quindi nuove armoniche. Lo stato dell'arte attuale tra gli algoritmi di Bass Enhancement è tuttavia un algoritmo non real-time proposto da *Moliner, Rämö e Välimäki* [6], che utilizza sia componenti non lineari sia l'analisi frequenziale tramite phase vocoder per processare separatamente transienti e toni.

4 | Risultati e confronto

Nel seguente capitolo sono analizzate le prestazioni ed i limiti degli algoritmi presentati al capitolo 2. I risultati ottenuti dagli algoritmi 2.1.1 e 2.1.2 sono coerenti con la teoria presentata.

4.1. Stimoli utilizzati

Per testare l'algoritmo sono stati utilizzati diversi segnali in ingresso. Qui sono presentati due esempi: il primo è il suono generato da un diapason, che può essere quindi considerato pseudo-stazionario; il secondo è una frase cantata, ovvero un segnale non stazionario con un'ampia variazione di pitch. I due file sono accessibili nella cartella `audio_samples` della repository.

4.1.1. Diapason

Nelle figure 4.1 e 4.2 sono mostrati rispettivamente lo spettrogramma e lo spettro di potenza del diapason del file `tuningfork.wav`.

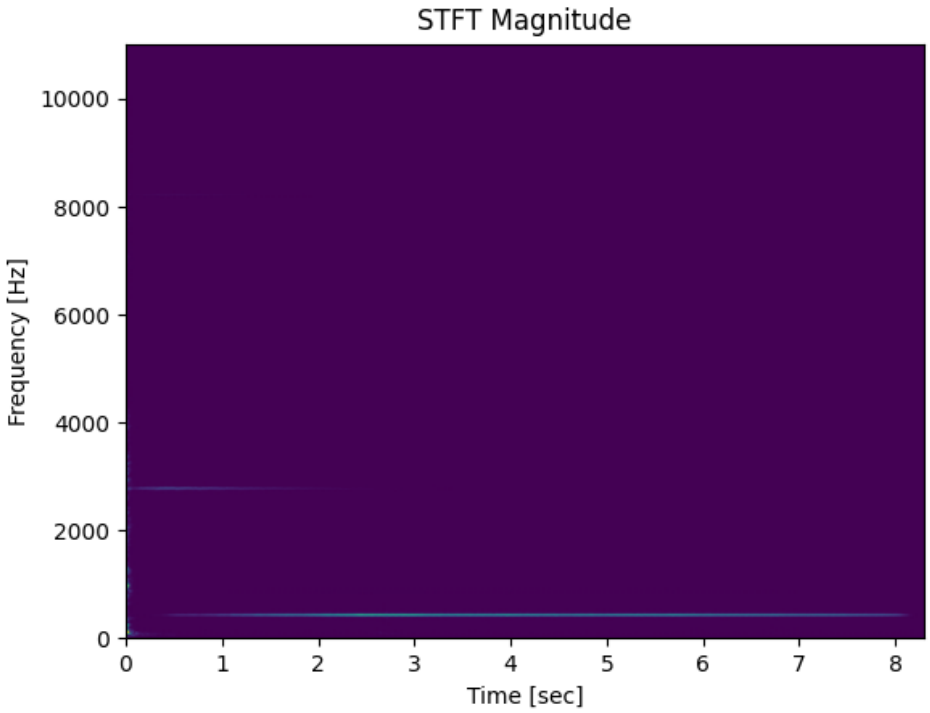

Figura 4.1: Spettrogramma del diapason.

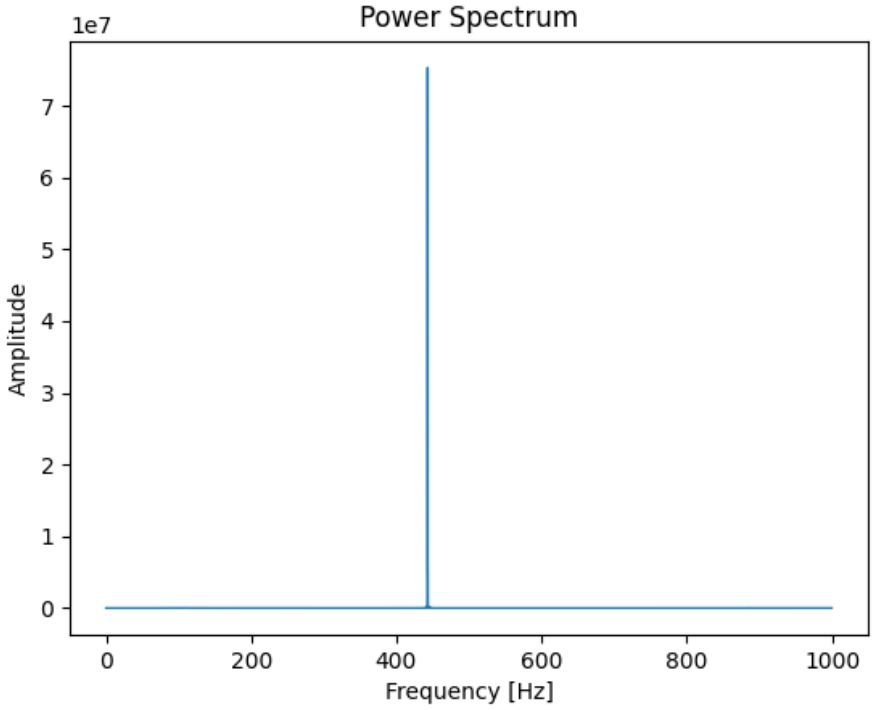

Figura 4.2: Spettro di potenza del diapason.

È possibile vedere chiaramente un'unica componente di frequenza tra i 440 ed i 450 Hz, come prevedibile trattandosi di un diapason.

4.1.2. Linea vocale

Nelle figure 4.3 e 4.4 sono mostrati rispettivamente lo spettrogramma e lo spettro di potenza della linea vocale del file `female_voice_1.wav`.

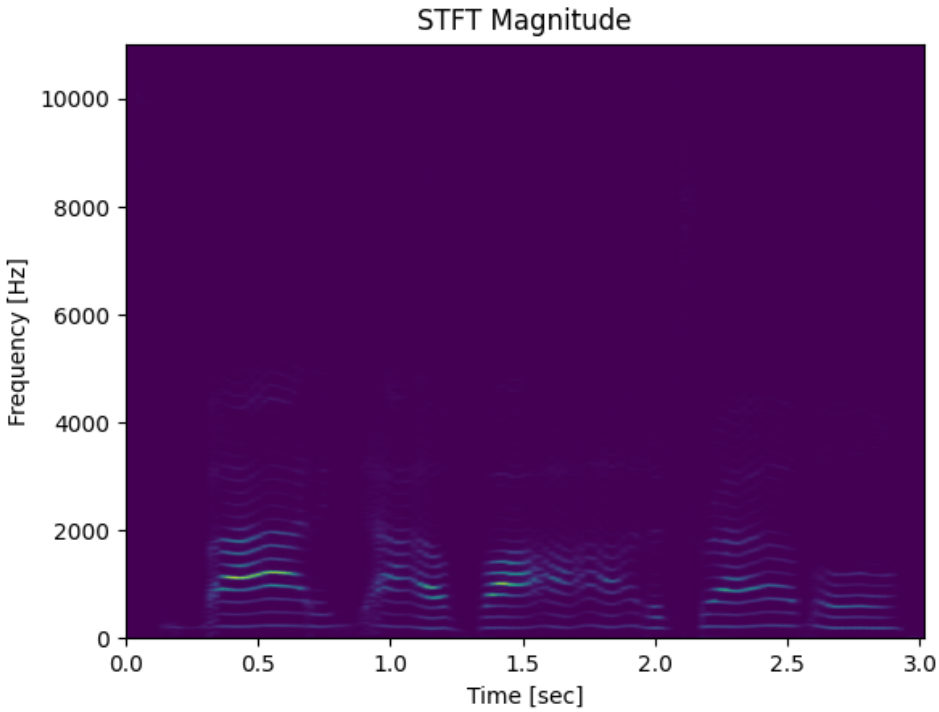

Figura 4.3: Spettrogramma della linea vocale.

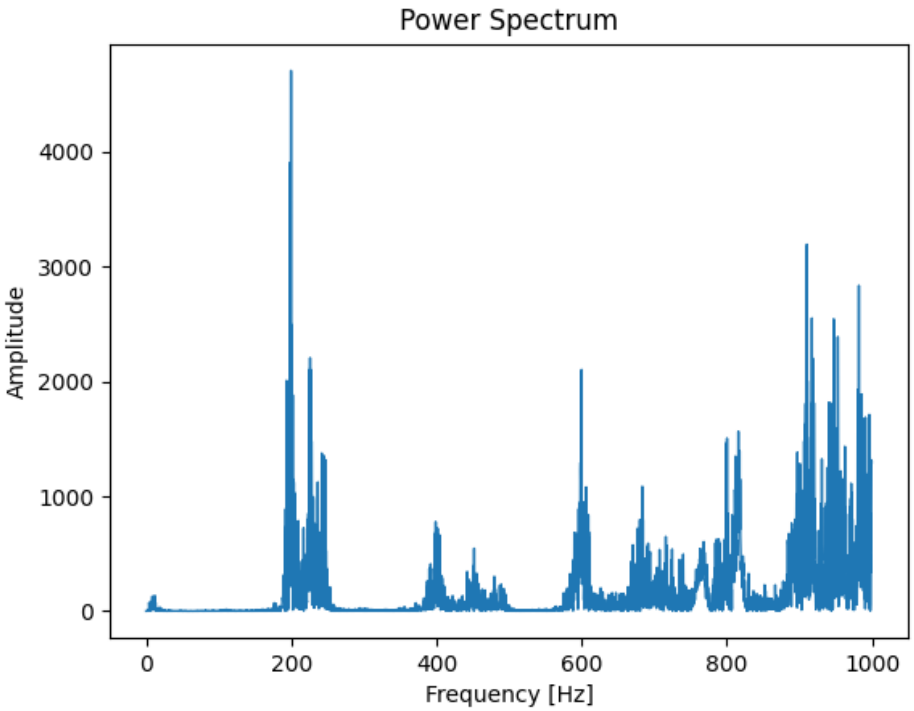

Figura 4.4: Spettro di potenza della linea vocale.

In questo caso, entrambe le figure mostrano una elevata complessità e ricchezza spettrale dovuta all'evoluzione del pitch nel tempo, essendo il segnale nè stazionario nè quasi-stazionario.

4.2. Algoritmo triviale

4.2.1. Prestazioni - diapason

Il pitch viene calcolato su tutta la lunghezza del segnale, pertanto l'output è un unico valore di pitch che viene stampato a terminale. In figura 4.5 è possibile vedere i log del terminale di Python.

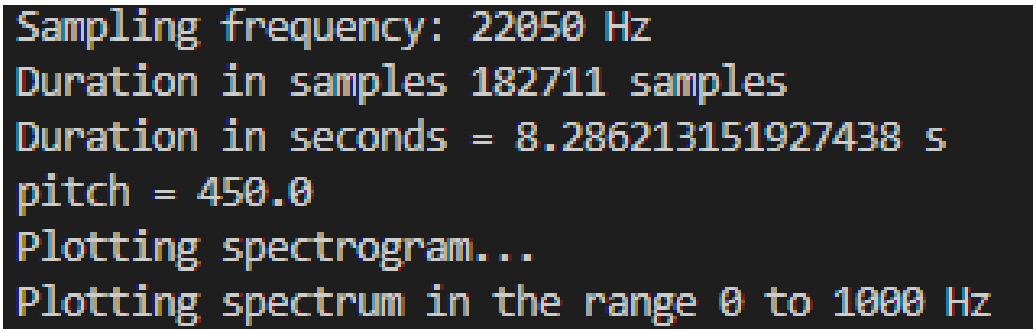

Figura 4.5: Alla riga 3 è possibile leggere il valore stimato del pitch.

Il pitch rilevato è pertanto molto vicino al pitch reale, con un errore poco superiore all' 1%, dovuto probabilmente solo alla quantizzazione.

4.2.2. Prestazioni - linea vocale

Stampare un unico valore di pitch non ha alcuna valenza in questo caso. Il pitch ottenuto corrisponde a 1050 Hz, che tuttavia non corrisponde a nessuna delle diverse componenti fondamentali del segnale. Questo test è quindi utile per dimostrare la necessità di un'analisi in short time dell'autocorrelazione.

4.3. Algoritmo short-time con pre-processing

I risultati ottenuti con l'algoritmo 2.1.2 sono direttamente confrontati con l'algoritmo YIN (sezione 2.2). I parametri utilizzati sono comuni per entrambi gli algoritmi, in modo da non presentare bias per una soluzione o l'altra e rendere quindi il confronto più robusto.

4.3.1. Prestazioni - diapason

In figura 4.6 è possibile notare come entrambe le soluzioni presentino una forte oscillazione all'inizio ed alla fine del segnale. Questo comportamento è probabilmente dovuto ad effetti di fade-in e fade-out nella traccia originale. Come prevedibile, questo fenomeno è molto più evidente nell'algoritmo presentato, non essendo immune ai cambiamenti di ampiezza. Al contrario, l'algoritmo YIN è più robusto rispetto a tali variazioni.

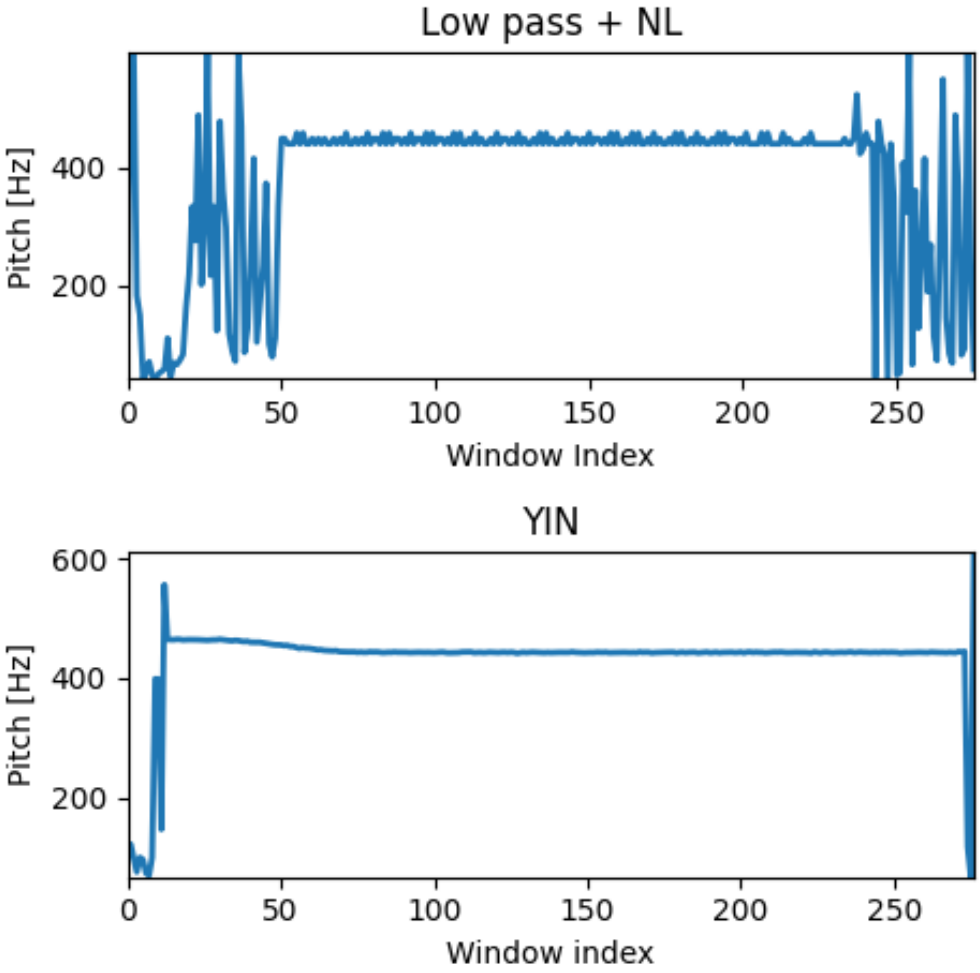

Figura 4.6: Pitch tracking del diapason. Sopra: algoritmo presentato. Sotto: algoritmo YIN.

Il corretto funzionamento dell'algoritmo è confermato dal comportamento nel range di indici $[50, 230]$, dove il pitch è rilevato correttamente intorno ai 445 HZ, con un oscillazione inevitabile di ± 5 Hz. Anche in questo caso, il risultato è coerente con la teoria presentata e l'errore è anche ben al di sotto del 10% teorizzato in tabella 2.1.

4.3.2. Prestazioni - linea vocale

Nel caso della linea vocale, è difficile stimare il pitch reale di ogni singola finestra. Si possono tuttavia confrontare gli andamenti del pitch tracciati dall'algoritmo proposto e dall'algoritmo YIN in figura 4.7. È quindi possibile notare che il rilevamento risulta corretto nel suo range di errore per la maggior parte del segnale.

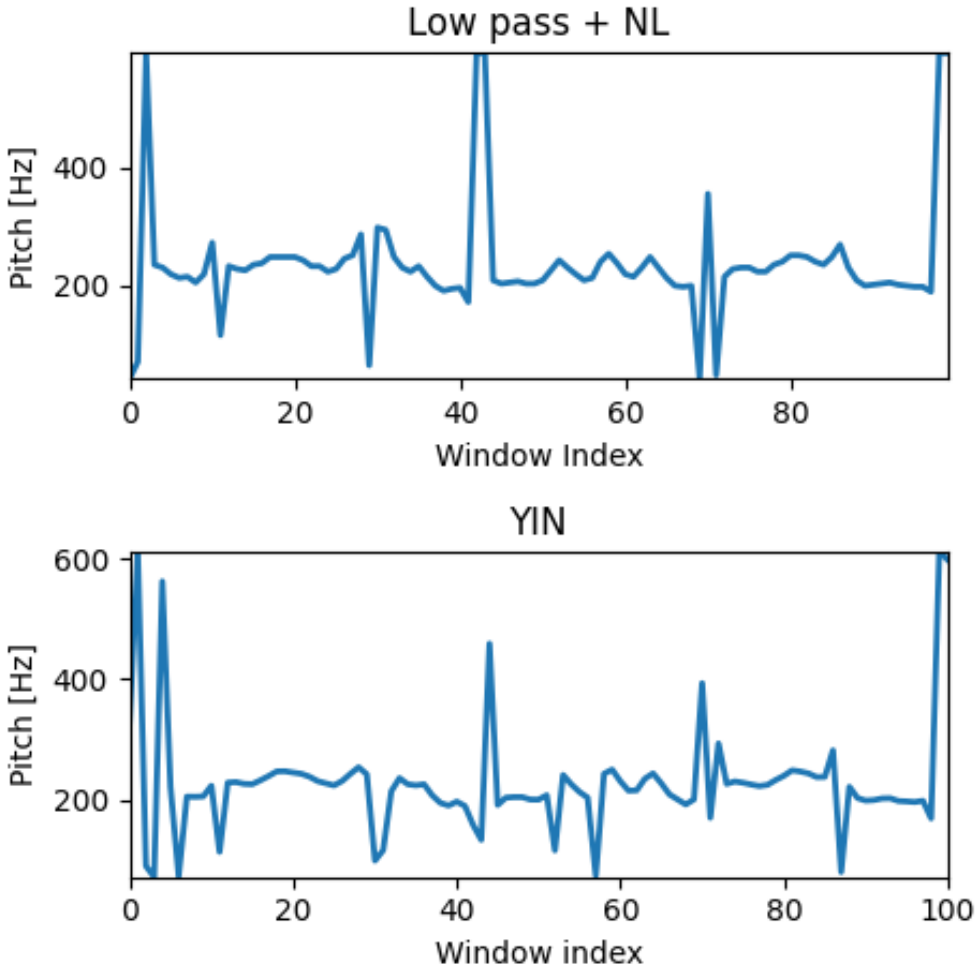

Figura 4.7: Pitch tracking della linea vocale. Sopra: algoritmo presentato. Sotto: algoritmo YIN.

Non mancano lievi artefatti, dovuti principalmente alla presenza di suoni senza pitch (fricativi o occlusivi, come spiegato in [3]) e silenzi. Questi, tuttavia, potrebbero essere risolti applicando un fine-tuning specifico ai parametri per il tipo di voce o melodia che si desidera analizzare. Non di meno, rimane un risultato soddisfacente per ottenere un'idea generale dell'andamento del pitch in una traccia audio.

4.4. Tracking della fondamentale mancante

Come spiegato al capitolo 3, il metodo dell'autocorrelazione è estremamente robusto per il rilevamento del pitch virtuale, ovvero in assenza di una componente di fondamentale nel segnale. In figura 4.8 è mostrato l'andamento del pitch della linea vocale già presentata,

alla quale è stato applicato un filtro passa alto con frequenza di taglio a 300 Hz con alta pendenza.

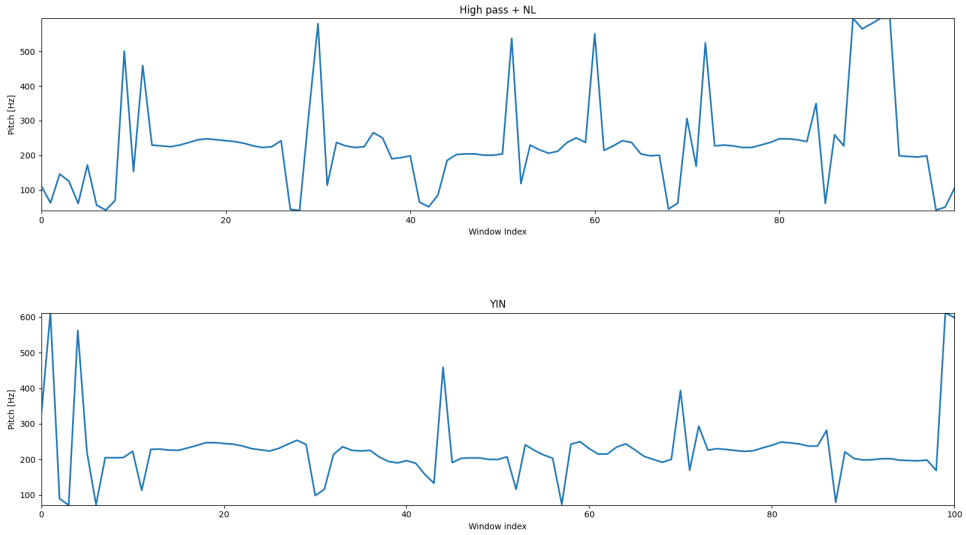

Figura 4.8: Sopra: algoritmo presentato. Sotto: algoritmo YIN.

Risulta evidente come siano rilevate curve di pitch al di sotto dei 300 Hz, nonostante fisicamente nel segnale tali frequenze non siano presenti, o siano molto attenuate. L'algoritmo YIN in particolare non soffre di alcun nuovo artefatto e traccia correttamente il pitch in ogni finestra, mentre l'algoritmo di correlazione diretta soffre della variazione in ampiezza dovuta al filtraggio, generando pitch virtuali spuri in alcune finestre. Il risultato è comunque da ritenersi soddisfacente, ed è prova che l'autocorrelazione rimane il miglior metodo per studiare il pitch in caso di fondamentale mancante.

Conclusioni

L'obiettivo dell'elaborato era di presentare le tecniche di rilevamento del pitch che utilizzano il metodo dell'autocorrelazione. Sono state discusse le basi teoriche della funzione di autocorrelazione ed il funzionamento generale degli algoritmi di Pitch Detection che sfruttano l'autocorrelazione. Sono state analizzate tre diverse soluzioni, una che applica il metodo diretto, una che applica una pre-elaborazione e lavora in short time, ed infine il metodo YIN. Delle prime due soluzioni, sono state presentate due implementazioni in ambiente Python, i cui risultati sono coerenti con le aspettative e con la teoria esposta nei primi capitoli. Infine, è stata spiegato il fenomeno del virtual pitch e la teoria della fondamentale mancante, portando esempi pratici di come sia comune nei suoni normali percepire un pitch che in realtà non è presente fisicamente nel segnale, e come questo venga sfruttato al giorno d'oggi anche nell'industria musicale e dell'elettronica di consumo. È stato inoltre dimostrato che i metodi basati sull'autocorrelazione siano decisamente robusti per il rilevamento del pitch in caso di fondamentale mancante.

Bibliografia

[1] A. de Cheveigne and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *J. Acoust. Soc*, 111, 2002.

[2] J. Goldstein. An optimum processor theory for the central formation of the pitch of complex tones. *J. Acoust. Soc.*, 54:1496–1516, 1973.

[3] M. Jiménez-Hernández. A tutorial to extract the pitch in speech signals using autocorrelation. *Open Journal of Technology & Engineering Disciplines (OJTED)*, 25: 01–10, 2016.

[4] E. Larsen and R. Aarts. *Audio bandwidth extension: Application of psychoacoustics, signal processing and loudspeaker design.* Wiley, 2004.

[5] G. Miller and W. Taylor. the perception of repeated bursts of noise. *J. Acoust. Soc.*, 20:171–182, 1948.

[6] E. Moliner, J. Rämö, and V. Välimäki. Virtual bass system with fuzzy separation of tones and transients. *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, 2020.

[7] L. Rabiner. On the use of autocorrelation analysis for pitch detection. *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING*, 25, 1977.

[8] E. Terhardt. Pitch, consonance, and harmony. *J. Acoust. Soc.*, page 1061–1068, 1974.

[9] E. Terhardt, G. Stoll, and M. Seewann. Pitch of complex signals according to virtual-pitch theory. *J. Acoust. Soc.*, 71, 1982.

[10] Wikipedia contributors. Missing fundamental — Wikipedia, the free encyclopedia, 2022. URL `https://en.wikipedia.org/w/index.php?title=Missing_fundamental&oldid=1084386414`. [Online; accessed 26-June-2022].