Estrazione di caratteristiche da segnali audio

Alessandro Avila

Abstract

Implementazione di un sistema che analizza i file mp3 in frequenza (analisi di Fourier) e che in base alle armoniche e potenza del segnale fa muovere il robot umanoide (con primitive di movimento già realizzate). Il lavoro si concentra sull'estrazione di informazioni quali ritmo e ampiezza del segnale, quindi sulla modulazione dei movimenti delle mani e delle braccia del robot.

1. MP3 - Definizione

MP3 *(per esteso **M**otion **P**icture Experts Group-1/2 Audio Layer **3**, ovvero MPEG-1 Audio Layer 3)* è una sigla utilizzata per indicare comunemente i file audio. Più precisamente è un algoritmo di compressione audio di tipo lossy. Scopo di tale algoritmo è quello di *comprimere* il più possibile un file audio, mantenendone, nei limiti del possibile, inalterata la qualità. In questo modo il file, una volta effettuata la codifica, raggiunge una dimensione (riferita in genere in MB) tale da poter essere velocemente trasferito su dispositivi periferici di memorizzazione secondaria (quali, per esempio, pendrive o schede di memoria) o facilmente trasmesso attraverso la rete.

L' MPEG (inteso come algoritmo) elimina dai file audio determinate informazioni che, sulla base di numerose ricerche in campo acustico, sono ritenute non necessarie.

Tali studi rivelano infatti che il nostro orecchio non è in grado di discernere tra frequenze cosiddette "forti" e frequenze "deboli", quando queste sono adiacenti o, in termini tecnici, quando le prime mascherano le ultime (vd. Figura 1 - *I segnali in rosso, indicati dalle frecce, mascherano i segnali tratteggiati, che non saranno percepiti dall' orecchio umano*).

La compressione messa in atto si propone appunto di eliminare le frequenze mascherate; ciò contribuirà a ridurre la dimensione del file finale.

1.1. Layers

Esistono vari livelli di compressione (Layers) utilizzati dall' MPEG. Tutti si basano sul medesimo schema di codifica per comprimere dati audio e sono legati all' algoritmo di prima generazione, si differenziano invece nella complessità del codice base. Qui di seguito ne viene fornita una breve descrizione.

1.1.1. Layer I. Usato nei sistemi digitali professionali. Adotta esclusivamente il metodo di eliminazione delle frequenze mascherate derivato dalle ricerche in campo acustico precedentemente menzionate. Significa che esso elimina quelle frequenze che vengono nascoste dietro ad altre più grandi.

1.1.2. Layer II. Adotta metodi di filtraggio del segnale audio molto più precisi rispetto al primo modello; è stato migliorato il metodo della scelta e della eliminazione delle frequenze non necessarie.

1.1.3. Layer III. Il layer III è il più complesso modello MPEG per l' audio. Adotta filtraggi più massicci rispetto al Layer II ma utilizza un codec estremamente complesso (Huffman).

2. Compressione

Occorre innanzitutto distinguere tra i vari tipi di compressione audio. Ne esistono principalmente due:

Senza perdita di dati: è un tipo di compressione (non necessariamente audio) che evita la perdita di informazione dai dati che si vogliono comprimere. Si tratta di un metodo adottato da programmi di archiviazione ,quali, ad esempio, WinZip o WinRar.

Con perdita di dati: compressione di tipo lossy, che implica la perdita di bit di informazione con lo scopo di ridurre la dimensione finale del file. Il formato MP3 sfrutta questo tipo di compressione.

Nella sezione seguente analizzeremo i differenti aspetti che entrano in gioco nella procedura di compressione degli MP3. Esamineremo quindi i processi tramite i quali un segnale audio viene digitalizzato ed archiviato sotto forma di file, da saranno estratte le informazioni necessarie al lavoro preso in esame.

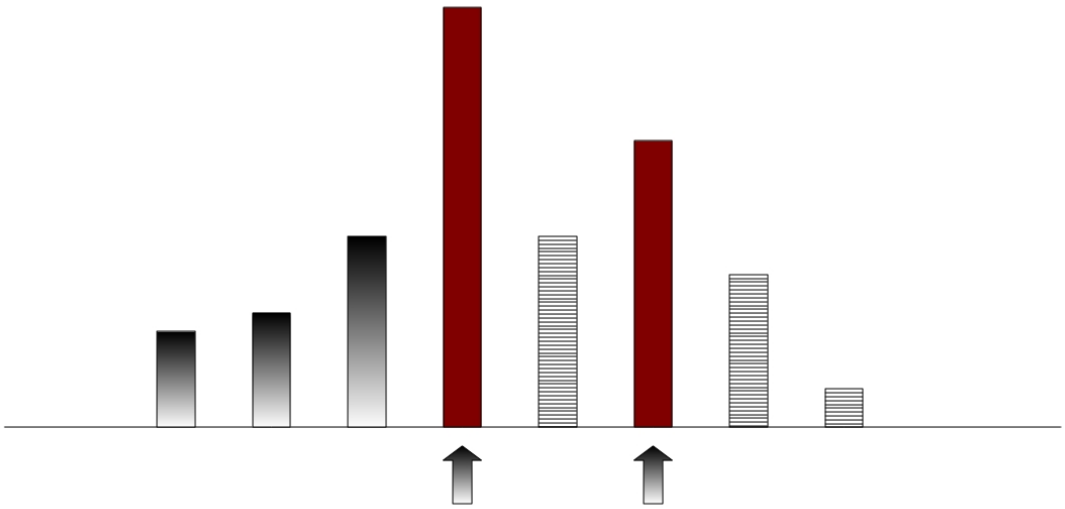


Figura 1. Segnali forti e segnali deboli

L'intera fase è suddivisa in cinque aspetti (*Campionamento*, *Effetto frequenze maschera*, *Effetto maschera temporale*, *Metodo di compressione dell' MPEG Layer III*, *MPEG AUDIO LAYERS*), dei quali solo il primo viene trattato, in quanto più pertinente con la discussione.

2.1. Campionamento

Con il termine campionamento si intende un processo di digitalizzazione ripetuta ad intervalli regolari nel tempo. Scopo del campionamento è quello di archiviare, in forma digitale, l'onda analogica, tramite la scomposizione del suono in un certo numero di informazioni al secondo (CAMPIONI).

Sono tre gli aspetti principali che intervengono in questa fase: **canale**, **frequenza di campionamento** e **bitrate.**

2.1.1. Canale. In telecomunicazioni il termine indica una via di comunicazione del segnale; in un' accezione più generale, con canale intendiamo una porzione di banda dello spettro elettromagnetico su cui veicolare segnali elettromagnetici.

Negli mp3 vi sono due canali: *mono* e *stereo*

2.1.2. Frequenza di campionamento. Misura (espressa in Hertz – Hz) che designa il numero di campioni presi al secondo (esempio: 44100 Hz = 44100 campioni al secondo).

In altri termini si tratta dello strumento con il quale il segnale audio di partenza, sotto forma di segnale analogico, viene misurato e tradotto in forma digitale (processo di trasformazione altrimenti noto come *discretizzazione*).

La codifica MPEG-1 utilizza spesso la frequenza di 44100 Hz per la creazione di MP3 (la stessa utilizzata nella codifica di CD-Audio).

Questa fase determina sensibilmente la risoluzione finale ottenuta dal processo di compressione.

2.1.3. Bitrate. Numero di unità binarie (bit), assegnate ai campioni (tipicamente 8-16 bit), che fluiscono al secondo, variabile per i file MP3. Il numero dei bit corrispondenti a ciascun campione determina di fatto l' intervallo dei possibili valori (8 bit

= 2^8 = 256 valori, 16 bit = 2^16 = 65536 valori) che danno informazioni sul livello relativo ad ognuno di essi.

Convertendo campioni da 16 bit a 8 bit si dimezza il file originario ma contemporaneamente si riduce pesantemente la qualità della musica.

La regola generale è che maggiore è il bitrate, più informazione è possibile includere dall' originale, maggiore è la qualità del file audio compresso. Attualmente le codifiche dei file MP3 fissano un tasso di compressione equivalente per tutto il file audio**.**

Per l' MPEG-1 Layer 3 i bitrate disponibili sono: 32, 40, 48, 64, 80, 96, 112, 128, 160, 192, 224, 256 e 320 kbit/s.

Il bitrate , assieme alla frequenza di campionamento, determina infine la qualità della codifica adottata, da cui dipenderà la qualità del file audio MP3, nonchè in generale l'efficienza dell' algoritmo di compressione. In Figura 2 uno schema generale.

3. Analisi in frequenza

Il lavoro si concentrerà sull' estrazione di informazioni caratteristiche del segnale audio tradotto in forma di file mp3. Una volta che l' analisi di Fourier fornirà i dati richiesti (l' attenzione è rivolta principalmente ad *ampiezza* e *ritmo* del segnale), concluderemo con la simulazione sul robot umanoide.

Affronteremo ed analizzeremo due modalità di estrazione; la prima si basa sulla libreria **Clam** di **Clam Project**, la seconda fa uso dei plugin della libreria **VAMP.**

4. Analisi 1

Si farà riferimento alla libreria C++ CLAM v.1.4.0 (http://clam-project.org). I contributi all' estrazione di ampiezza e ritmo del segnale sono rispettivamente nei file (1) e (2), discussi nelle due sezioni successive:

(1) AudioFileReading_example.cxx

(2) TickExtractor.cxx

4.1. Ampiezza

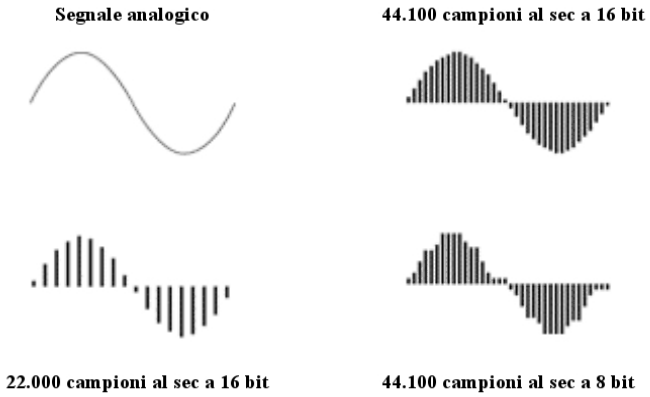


Figura 2. Schema del campionamento

Il file (1) implementa una sorta di **VU-meter**, un

misuratore che indica la rumorosità (**Loudness**) di un segnale dato.

Questa "loudness" viene calcolata per tutti i campioni di cui è composto il file mp3. Il calcolo restituisce un valore compreso nell' intervallo [0, -inf], in scala dB (decibel). Il codice, semplicemente, mantiene un contatore che memorizza, ad ogni ciclo, il massimo valore di loudness ottenuto per il generico campione nel canale (mono o stereo).

Al termine, il programma restituisce l'ampiezza del segnale (vd. Figura 3).

Prima di procedere, occorre tuttavia dare una definizione di loudness, caratteristica distintiva di questa analisi in frequenza.

4.1.1. Loudness. La loudness (o sound loudness) è la caratteristica di un suono soggettivamente correlata all' ampiezza dell' onda. In altri termini è definita come "quell' attributo di sensazione uditiva in base al quale i suoni possono essere disposti in una scala che va da 'calmo' a 'rumoroso'" (cit. American National Standards Institute, "American national psychoacoustical terminology" S3.20, 1973, American Standards Association). La loudness è perciò un termine soggettivo che descrive l' intensità di un suono in termini della sua percezione da parte dell' orecchio umano. Tale intensità deve quindi essere calcolata tenendo conto della sensibilità dell' orecchio a quella particolare frequenza. Da considerare inoltre che la risposta dell' orecchio all' aumentare dell' intensità del suono è un "fattore logaritmico"; motivo per cui si utilizza una scala in decibel (dB) per poterla stimare (vd. Figura 4).

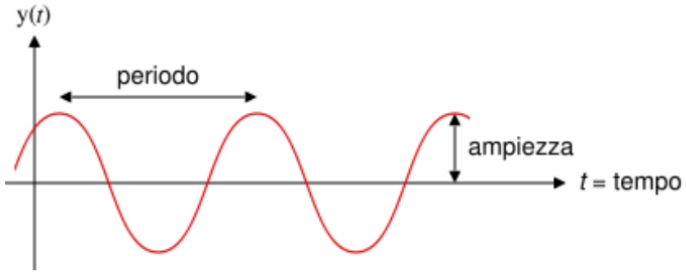
Occorre tuttavia precisare che questa è una misura indicativa, in quanto la percezione della loudness varia da soggetto a soggetto e non è possibile darne una definizione oggettiva.

La loudness è infine influenzata da parametri fisici dell' onda quali: *frequenza*, *ampiezza di banda* e *durata*. In Figura 5 un altro grafico.

4.2. Ritmo

Il file (2) produce un file XML contenente i dettagli sulle 3 caratteristiche principali relative al ritmo del segnale: **onset**, **beats** e **ticks**. Diamone una definizione:

- ONSET: Si riferisce all' istante di tempo (onsets per più istanti) in cui ha inizio un evento musicale.



Ampiezza dell' onda di equazione: $y = A\ sin(t - K) + b$
- A: l'ampiezza dell'onda
- y: la variabile che oscilla
- K e b: costanti arbitrarie che rappresentano rispettivamente gli offset di tempo e spazio

Figura 3. Ampiezza

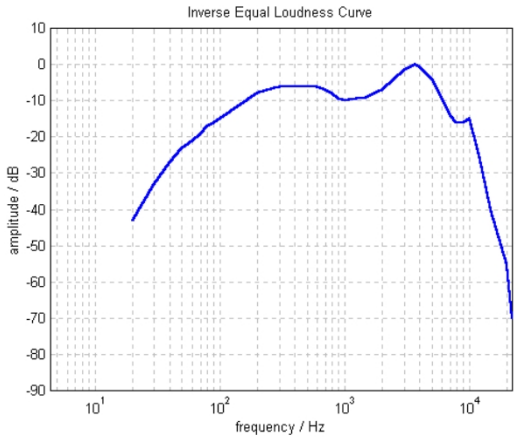


Figura 4. Curva inversa loudness

- BEATS: comunemente noti come “battiti” (battiti per minuto). I battiti per minuto (bpm – beats per minute) definiscono propriamente il ritmo del segnale, in quanto indicano il numero di note (le note nere in un pentagramma musicale) che si susseguono a cadenza regolare.

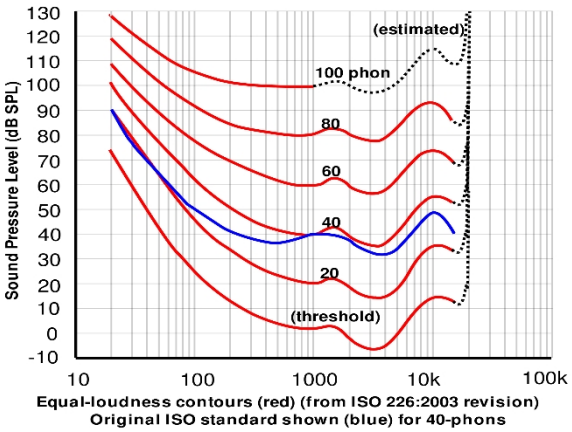


Figura 5. Curve loudness

- TICK: minimo intervallo di tempo in cui le note sono quantizzate.

- METER: tipo di compasso (4/4, 3/4, …).

5. Analisi 2

In questa sezione si farà uso dello strumento "Sonic Annotator" (http://omras2.org/SonicAnnotator) e dei dei plugin della libreria VAMP (si veda a tal proposito

la Figura 6 per uno schema generale).

5.1 Sonic Annotator

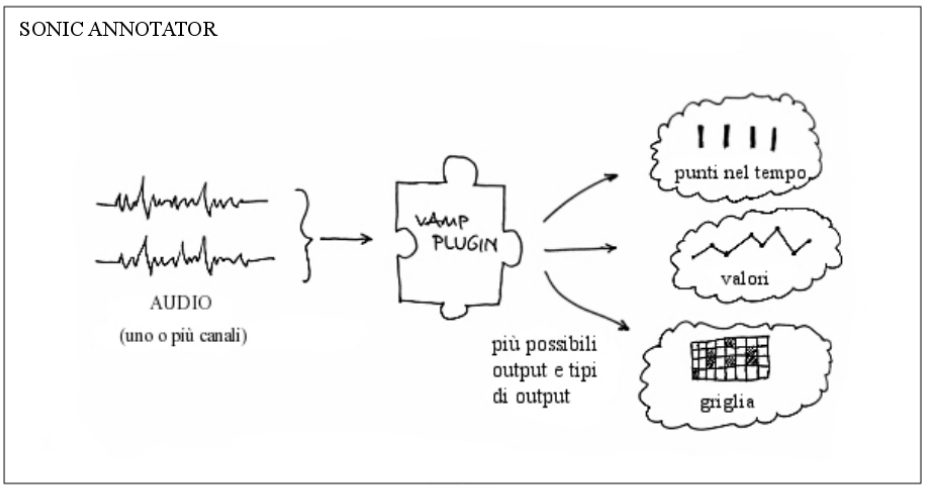


Figura 6. Host, schema generale

Il sonic annotator è un programma "da linea di comando", interamente scritto in C++, che ha la funzione di *estrazione batch* di caratteristiche audio a partire da uno o più file (vari formati sono accettati: aiff, au, avr, caf, flac, htk, iff, mat, **mp3**, ogg, paf, pvf, raw, sd2, sds, sf, voc, w64, wav, xi).

Elemento fondamentale, inoltre, è la possibilità di gestire, modificare le modalità di estrazione di tali caratteristiche, facendo uso dei cosiddetti VAMP Plugins.

I formati di output sono molteplici, e possono esserne definiti di nuovi; quelli standard sono: RDF (Resource Description Format) e CSV (Comma-Separated Values).

Sono necessari alcuni file per poter eseguire il programma, da passare come argomenti allo stesso nel seguente ordine:

- **Una collezione di file audio**. Per i nostri scopi si
- farà l' analisi di un solo file alla volta.
- Il plugin o i **plugin VAMP** da utilizzare per l'analisi.
- **Un file RDF** per ciascun plugin che ne descriva i parametri interni richiesti per l'estrazione.
- Il tipo di **output** desiderato.

5.2. Estrazione

In questa sezione sarà descritta la procedura di estrazione delle informazioni desiderate.

Da linea di comando verrà invocato il programma sonic annotator che utilizzeremo per visualizzare anzitutto i plugin installati:

```
$ sonic-annotator -l

vamp:vamp-example-plugins:amplitudefollower:amplitude
vamp:vamp-example-plugins:fixedtempo:acf
vamp:vamp-example-plugins:fixedtempo:detectionfunction
vamp:vamp-example-plugins:fixedtempo:filtered_acf
vamp:vamp-example-plugins:fixedtempo:tempo
vamp:vamp-example-plugins:fixedtempo:candidates
vamp:vamp-example-
plugins:percussiononsets:detectionfunction
vamp:vamp-example-plugins:percussiononsets:onsets
vamp:vamp-example-plugins:powerspectrum:powerspectrum
vamp:vamp-example-plugins:spectralcentroid:linearcentroid
vamp:vamp-example-plugins:spectralcentroid:logcentroid
vamp:vamp-example-plugins:zerocrossing:counts
vamp:vamp-example-plugins:zerocrossing:zerocrossings
```

Faremo uso esclusivamente dei plugins: amplitudefollower e fixedtempo.

5.2.1. Amplitude follower. Tiene traccia e produce in uscita l'ampiezza del segnale audio campione per campione, ritornando i picchi blocco per blocco.

Calcola strutturalmente semplici dati bidimensionali (tempo e valore) dall' audio, per poter essere stampati a video od usati per altri propositi.

Per visualizzare le caratteristiche e i parametri del plugin, da linea di comando:

```
$ sonic-annotator -s vamp:vamp-example-
plugins:amplitudefollower:amplitude

@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix vamp:   <http://purl.org/ontology/vamp/> .
@prefix :       <#> .

:transform_plugin a vamp:Plugin ;
  vamp:identifier "amplitudefollower" .

:transform_library a vamp:PluginLibrary ;
  vamp:identifier "vamp-example-plugins" ;
  vamp:available_plugin :transform_plugin .

:transform a vamp:Transform ;
  vamp:plugin :transform_plugin ;
  vamp:step_size "1024"^^xsd:int ;
  vamp:block_size "1024"^^xsd:int ;
  vamp:parameter_binding [
    vamp:parameter [ vamp:identifier "attack" ] ;
    vamp:value "0.01"^^xsd:float ;
  ] ;
  vamp:parameter_binding [
    vamp:parameter [ vamp:identifier "release" ] ;
    vamp:value "0.01"^^xsd:float ;
  ] ;
  vamp:output [ vamp:identifier "amplitude" ] .
```

5.2.1. Fixed Tempo. Semplicemente analizza un frammento di audio e ne analizza i battiti per minuto (bpm). Come si evince dal successivo output, i parametri del plugin sono i seguenti:

- **Tempo stimato minimo, tempo stimato massimo (bpm)**: questi parametri controllano il range di valori all' interno del quale il plugin fixed tempo produrrà la sua stima.
- **Dimensione della finestra temporale (secondi)**: indica in secondi la durata del frammento audio su cui calcolare i bpm.

```
$ sonic-annotator -s vamp:vamp-example-
plugins:fixedtempo:tempo

@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix vamp:   <http://purl.org/ontology/vamp/> .
@prefix :       <#> .

:transform_plugin a vamp:Plugin ;
  vamp:identifier "fixedtempo" .

:transform_library a vamp:PluginLibrary ;
  vamp:identifier "vamp-example-plugins" ;
  vamp:available_plugin :transform_plugin .

:transform a vamp:Transform ;
  vamp:plugin :transform_plugin ;
  vamp:step_size "64"^^xsd:int ;
  vamp:block_size "256"^^xsd:int ;
  vamp:parameter_binding [
    vamp:parameter [ vamp:identifier "maxbpm" ] ;
    vamp:value "190"^^xsd:float ;
  ] ;
  vamp:parameter_binding [
    vamp:parameter [ vamp:identifier "maxdflen" ] ;
    vamp:value "10"^^xsd:float ;
  ] ;
  vamp:parameter_binding [
    vamp:parameter [ vamp:identifier "minbpm" ] ;
    vamp:value "50"^^xsd:float ;
  ] ;
  vamp:output [ vamp:identifier "tempo" ] .
```

Come è facile vedere, i valori di default di questi tre parametri sono, rispettivamente:

- Tempo stimato massimo - “maxbpm”: 190 bpm
- Tempo stimato minimo - “minbpm”: 50 bpm
- Dimensione della finestra temporale - “maxdflen”: 10 s

Stamparemo un esempio in cui estrarremo in tempo reale, a partire da un file mp3 in input, la sua ampiezza e il suo ritmo.

Codice:

```
$ sonic-annotator -t vamp-example-
plugins:amplitudefollower:amplitude.n3 file.mp3 -w csv
--csv-stdout &&
```

```
sonic-annotator -t vamp-example-
plugins:fixedtempo:tempo.n3 file.mp3 -w csv --csv-stdout
```

(lo switch -t richiede un file rdf che contiene la struttura del plugin interessato, nel caso di specie i due file .n3 contengono rispettivamente il primo ed il secondo output)
(lo switch -w richiede di specificare il tipo di output, in questo caso si è scelto la semantica 'csv')

```
0.023219954, 0.10403
0.046439909, 0.119019
0.069659863, 0.115642
0.092879818, 0.10713
0.116099773, 0.104197
0.139319727, 0.0949763

[ … ]

198.646712018, 5.53106e-25
198.669931972, 2.63524e-27
198.693151927, 1.25555e-29
198.716371882, 5.98197e-32
198.739591836, 2.85007e-34

0.000000000, 49.999818594, 149.796, "149,8 bpm”
```

6. Robot umanoide NAO

Prodotto dalla parigina *Aldebaran Robotics*, il NAO (Figura 7) si propone essere il prodotto di punta nella famiglia di robot umanoidi; l'azienda produttrice ha infatti annunciato di voler produrre il drone su larga scala nel 2010.

Il robot NAO si distingue innanzitutto per una importante caratteristica: il comportamento del dispositivo può essere completamente personalizzato, grazie al programma *Choregraphe* che fa da interfaccia per la modulazione dei movimenti meccanici.

Qui di seguito una tabella riassuntiva con le principali specifiche tecniche (Tabella 1).

CARATTERISTICHE TECNICHE	
Altezza	58 cm
Peso	4.3 kg
Materiale	Plastica
ENERGIA	
Alimentatore	AC 90-230 volts DC 24 volts
Autonomia	90 min
GRADI DI LIBERTÀ (DOF)	
Testa	2 DOF
Braccio	5 DOF ciascuno
Bacino	1 DOF
Leg	5 DOF ciascuno
Mano	1 DOF ciascuno
MULTIMEDIA	
Altoparlanti	2
Microfoni	4
Vista	Due videocamere CMOS con una risoluzione di 640x480 pixel, capaci entrambe di catturare sino a 30 immagini al secondo.
ACCESSO ALLA RETE	
Tipi di connessione	Wi-fi (IEEE 802.11g)
	Connessione Ethernet
ATTUATORI	
Il design originale Aldebaran RoboticsTM si basa su:	Sensori ad effetto Hall
	Microcontrollori dsPICS
	Motori coreless MAXON
SENSORI	
Tipo	32 sensori a effetto Hall
	1 sensore giroscopico a 2 assi
	1 accelerometro a 3 assi
	2 paraurti
	2 sonar
	2 I/R

	1 sensore tattile
LED	
Sensore tattile	12 LED 16 Blue levels
Occhi	2 x 8 LED RGB Fullcolour
Orecchie	2 x 10 LED 16 Blue levels
Busto	1 LED RGB Fullcolor
Piedi	2 x 1 LED RGB Fullcolor
SCHEDA MADRE	
x86 AMD GEODE 500MHz CPU	256 MB SDRAM 2 GB memoria flash
SOFTWARE EMBEDDED	
OS	Linux Embedded (32 bit x86 ELF) che usa distribuzione OpenEmbedded
Linguaggi di programmazione	C, C++, URBI, Python, .Net

Tabella 1. Caratteristiche tecniche