POLITECNICO DI MILANO
Corso di Laurea MAGISTRALE in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

UN MODELLO DEL CIRCUITO CORTECCIA-GANGLI-TALAMO PER LA GENERAZIONE DEL MOVIMENTO A PARTIRE DA PRIMITIVE MOTORIE

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof.ssa Giuseppina Gini
Correlatore: Ing. Alessio Mauro Franchi

Tesi di Laurea di:
Danilo Attuario, matricola 781526

Anno Accademico 2013-2014

Sommario

Uno degli obiettivi principali della robotica cognitiva è quello di creare robot che abbiano cognizione, cosi da poter interagire in modo naturale e intelligente con l'uomo. La capacità di movimento in un contesto come quello che l'uomo affronta ogni giorno risulta una abilità necessaria di cui questi sistemi devono essere dotati. Di recente sono stati sviluppati robot antropomorfi e umanoidi che hanno posto importanti sfide e portato nuove opportunità. Sono strutture molto complesse, con un alto numero di gradi di libertà, e le tecniche usate in passato per il loro controllo non sono piu' applicabili. Si ricercano nuovi metodi per sfruttare le loro potenzialità sempre più vicine a quelle umane. Lo scopo di questa tesi è la creazione di un modello che simuli il circuito corteccia-gangli-talamo alla base del cervello per la generazione del movimento in questi nuovi robot. Abbiamo sviluppato un sistema biologicamente ispirato per l'apprendimento e il controllo del movimento basato sull'idea delle primitive motorie. Gli esperimenti sono stati eseguiti seguendo come caso d'uso il robot antropomorfo NAO, con l'obiettivo di testare la sua capacità di generazione del movimento in un gioco come quello di coprire con un bicchiere una pallina posta su un tavolo. I risultati mostrano come il sistema sia una buona base per la creazione di un'architettura motoria, che ha una naturale integrazione con un sistema cognitivo di codifica degli stimoli e generazione di nuovi obiettivi al fine di permettere l'apprendimento senso-motorio in robot autonomi.

Abstract

The aim of cognitive robotics is to create robots that show cognitive capabilities such that they are able to interact in a natural and intelligent way with humans. Movement skills in the context where human beings live every day are fundamentals skills that these systems must exhibit. With the advent of anthropomorphic and humanoid robots a large number of new challenges and opportunities have been posed. These bodies are very complex, they have a high number of degree of freedom and the methods used since now to control them are no longer efficient. The purpose of this work is to create a system that can copy with these challenges and exploit robots potentialities that are more and more similar to human ones. We developed a biosipired model of cortex-basal ganglia-thalamus circuit in the brain for movement generation in these new robots. Our model is able to learn and control movements starting from a set of motor primitives. Experiments were carried out using the anthropomorphic robot NAO with the purpose of testing its ability in generating movements in a game where NAO had to cover a ball on a table with a glass. Results show that the system can be a good starting point for the creation of a more complex motor system, that has a natural integration with a cognitive architecture of sensory processing and self-generating goals to define a bioispired sensorimotor system.

Indice

Elenco delle figure

Capitolo 1

Introduzione

Negli ultimi anni stiamo assistendo a grandi progressi nel campo della robotica con la creazione di nuovi automi leggeri, complessi e dotati di un numero elevato di gradi di libertà. Lo scopo della progettazione di questi sistemi è, tra i tanti, avvicinarsi sempre più ad una natuale coesistenza con l'uomo nella società. I robot dovranno quindi affrontare situazioni nelle quali l'uomo si trova ogni giorno, mostrando un comportamento intelligente che permetta loro di interagire con un ambiente che ha una dinamica complessa, mostrando autonomia e capacità di apprendimento in tempi brevi.
Molteplici sono le situazioni e le opportunità di impiego di questi robot, così come le sfide che pongono. Una tra queste è la capacità di movimento che è una caratteristica fondamentale degli esseri intelligenti. Il movimento permette di interagire con l'ambiente circostante ed è al tempo stesso un mezzo per conoscerlo, per sviluppare nuovi obiettivi e per soddisfarli. Lo scopo di questa tesi è dare un contributo alla ricerca di una soluzione che permetta a robot complessi di muoversi, che è un probema dove generalmente i metodi di controllo classici fanno fatica o addirittura falliscono per via dell'altro numero di gradi di libertà e della complessità di elaborazione di un ricco ambiente sensoriale. Cerchiamo la soluzione ispirandoci alla natura, con l'obiettivo di progettare un modello che imiti il comportamento dei sistemi biologici che risolvono questo problema ogni giorno esibendo straordinarie doti di flessibilità e velocità.
Il lavoro svolto è lo sviluppo di un modello di generazione del movimento che ricrea il circuito naturale interno al cervello tra la corteccia, i gangi alla base e il talamo. Abbiamo individuato e studiato i fondamenti neurobiologici dei processi di codifica degli stimoli e di generazione del movimento, ed è emerso che le aree della corteccia, dei gangli alla base e del talamo formano un circuito di elaborazione che permettono all'uomo di muoversi.

Ci siamo concentrati sulla modellizzazione della corteccia e dei gangli, tralasciando l'area del talamo che svolge una funzione di collegamento e scambio di informazioni tra le altre due aree. La soluzione proposta realizza un sistema di generazione del movimento basato sulle primitive motorie, che sono movimenti semplici stereotipati usati come elementi base per costruire un movimento complesso. Il modello si compone di due macroaree: una dedicata alla rappresentazione dello stato e dell'interesse che il robot ha per esso e l'altra alla costruzione del movimento. Quest'ultima è a sua volta composta da un modulo per combinare le primitive motorie che corrisponde all'area della corteccia motoria, un modulo di apprendimento per rinforzo del primo modulo che corrisponde all'area dei gangli e un dataset di primitive disponibile al robot, sia innate che acquisite. La prima macroarea è stata ricavata da una architettura bioispirata esistente, chiamata IDRA, che modellizza l'area della corteccia sensoriale replicando il suo meccanismo di riduzione della dimensionalità dell'input per la creazione di una rappresentazione compatta dello stato, condivisa tra le altre aree del cervello [1]. Inoltre associa ad ogni stato un segnale di interesse che può essere di origine innata o sviluppato con l'esperienza. Questo segnale è utile per collegare l'aspetto cognitivo di IDRA con il nostro modello motorio, che è una delle motivazioni che hanno spinto questo lavoro di tesi. Il modello è stato implementato e infine testato su un robot umanoide reale per valutarne la capacità di generazione del movimento, prima in una versione senza IDRA, poi con i due sistemi integrati. I risultati hanno mostrato la capacità di apprendimento e generazione del movimento del nostro modello e sono incoraggianti per altri sviluppi futuri che si propongono di completare il lavoro proposto con la modellizzazione di altre aree del cervello, come il cervelletto.

Il concetto chiave su cui si basa il modello proposto è quello di primitiva motoria. Esistono molti studi che provano l'esistenza di questo elemento nell'uomo e tutti individuano in esso una causa di riduzione di dimensionalità che semplifica il controllo e l'apprendimento motorio [2] [3]. Attraverso l'osservazione dei segnali EMG effettuata su diversi campioni di individui, si è osservata la presenza di queste primitive, anche chiamate muscle synergies, e come le attività più complesse vengono generate da una combinazione di questi moduli base, che possono essere innati o appresi con l'esperienza. Altre teorie si spingono oltre e hanno portato ad affermare che il sistema nervoso centrale memorizza e combina le primitive in base al compito che esse realizzano [4]. Ciò significa che ogni primitiva ha un suo scopo, ovvero controlla l'andamento di una variabile nel tempo come può essere la distanza tra la mano e un oggetto nelle azioni di raggiungimento. Quest'ul-

tima definizione di primitiva costituisce l'idea su cui abbiamo sviluppato il modello della corteccia motoria, che genera il movimento a partire da una combinazione di gruppi di primitive che hanno uno stesso scopo.
Molte sono anche le formulazioni matematiche che formalizzano le primitive [5] [6] [4]. Di questi modelli ne abbiamo scelto uno, secondo un criterio che accompagna le nostre scelte di progettazione per tutto il lavoro svolto, ovvero implementiamo la soluzione che risulta il più possibile bioispirata. Il modello scelto è quello delle Dynamic Movement Primitives, o DMP, che fornisce una rappresentazione in termini cinematici delle primitive con importanti proprietà che rendono semplice combinarle e apprenderle. Le DMP costituiscono il dataset di primitive a disposizione dei vari moduli del nostro modello.
Osservando la struttura anatomica dei gangli alla base e sfruttando le proprietà delle DMP costruiamo il modulo di apprendimento per rinforzo implementando un algoritmo Actor-Critic di ispirazione biologica [7]. Questo è il modulo più importante nel modello proposto perchè ha il compito critico dell'apprendimento motorio.
Il lavoro di tesi svolto propone un modello di generazione del movimento biospirato per indirizzare il problema del movimento in robot complessi nell'ottica di una integrazione con funzionalità cognitive che permette l'esistenza di robot intelligenti autonomi. Per questo motivo abbiamo integrato e testato il nostro modello con l'architettura IDRA.

La tesi è strutturata nel modo seguente.
Nel **capitolo due** si illustrano alcune teorie neurobiologiche che sono alla base dei processi collegati alla generazione del movimento.
Nel **capitolo tre** si approfondiscono i motivi che ci hanno spinto ad intraprendere questa ricerca.
Nel **capitolo quattro** si mostra il modello che abbiamo sviluppato e il suo funzionamento senza addentrarci nei particolari implementativi.
Nel **capitolo cinque** si illustrano gli aspetti prettamente matematici alla base dell'architettura propostra.
Nel **capitolo sei** si mostrano i risultati sperimentali dei test eseguiti con il robot.
Nel **capitolo sette** vengono, in conclusione, evidenziati gli obiettivi di questo lavoro che sono stati raggiunti, proponendo inoltre alcuni interessanti possibili sviluppi futuri.

Capitolo 2

Modelli neurobiologici del movimento

In questo capitolo si indagano alcune teorie neurobiologiche che sono alla base dei processi collegati alla generazione del movimento. Si parte da come il cervello codifica gli stimoli sensoriali per poi passare a come affronta il problema della composizione dei comandi relativi all'azione che vogliamo compiere. Quando il sistema nervoso centrale genera movimenti volontari molti muscoli sono attivati e coordinati, ognuno composto da migliaia di unità motorie. E' un calcolo computazionalmente complesso e si è ipotizzato una organizzazione modulare del movimento. Presentiamo quindi una panoramica dei principali modelli trovati in letteratura per questi moduli dedicando inoltre un paragrafo per il modello adottato in questo lavoro di tesi. Presentiamo infine i paradigmi di apprendimento automatico che verranno implementati nella nostra architettura, come essi sono implementati dal circuito corteccia-gangli-talamo studiato.

2.1 Il cervello e il sistema senso-motorio

L'uomo gestisce compiti molto complessi come quello del movimento in maniera veloce e flessibile rispetto alle soluzioni fino ad ora progettate per i sistemi artificiali. Si vuole quindi prendere ispirazione dalle soluzioni adottate dall'uomo facendo luce sui processi che avvengono nel cervello.

2.1.1 Reazioni agli stimoli

Il cervello umano è composto da circa cento miliardi di cellule nervose e molte più cellule ausiliarie. Esistono diverse tipologie di queste cellule e la

loro composizione in strutture anatomiche differenti da luogo ad una suddivisione del cervello in aree specializzate nello svolgimento dei diversi processi che sono alla base dell'attività mentale.

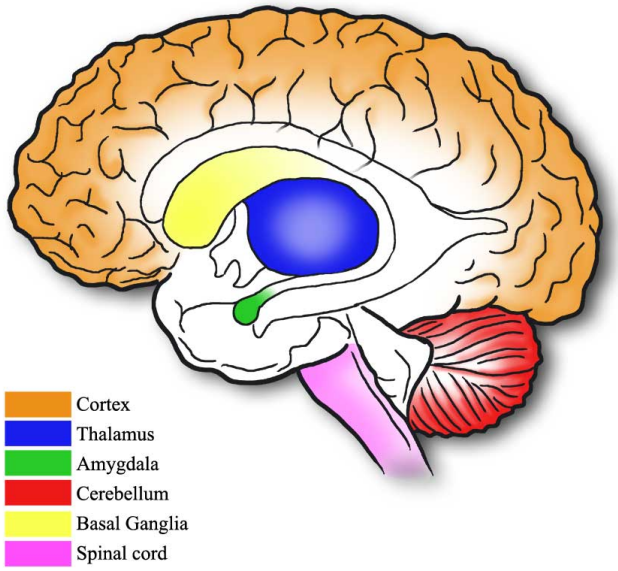

Figura 2.1: Aree di interesse del cervello.

Le aree sono fortemente interconnesse e si sono osservati dei percorsi di scambio di informazione che individuano dei sottosistemi con funzionalità specifiche. Ad esempio, il circuito amigdala-talamo-corteccia è di notevole importanza nello sviluppo cognitivo dell'essere umano perchè permette lo sviluppo di nuovi obiettivi sulla base dell'esperienza e di istinti, emozioni e paure innate [8]. Un altro circuito fondamentale all'attività umana è quello corteccia-gangli alla base-cervelletto-talamo che svolge buona parte dei processi mentali necessari all'apprendimento e alla esecuzione di varie tipologie di movimenti [7]. In particolare, un sottocircuito del precendente, composto dalla corteccia-gangli e talamo, rapprensenta l'oggetto di studio del presente lavoro di tesi.
Il primo passo nell'interazione con l'ambiente circostante è la reazione agli stimoli esterni. I nostri sistemi sensoriali elaborano in maniera simile segnali di modalità differente quali ad esempio visiva, olfattiva o acustica. I recettori sensoriali rispondono a particolari aspetti dello stimolo operando un filtraggio del segnale iniziale. Le diverse rappresentazioni vengono poi propagate

con un meccanismo simile a quello delle sinapsi fino alla corteccia sensoriale. Le regioni corticali in corrispondenza delle quali sono rappresentate modalità sensoriali differenti comunicano tramite collegamenti intracorticali con delle regioni associative multimodali, cioè quelle regioni che selezionano e integrano i segnali che ricevono dalle differenti regioni unimodali [9].
Le ricerche sulla visione, ad esempio, hanno dimostrato che le informazioni visive arrivano dalla retina al cervello attraverso almeno due vie parallele distinte: la via M e la via P. La separazione delle informazioni visive inizia già nella retina grazie a due diverse tipologie di popolazione di cellule gangliari costituite, rispettivamente, da cellule grandi (cellule M) e cellule piccole (cellule P). Queste due tipologie cellulari portano informazioni alquanto differenti a diversi strati del talamo: le prime proiettano agli strati magnocellulari del corpo genicolato laterale del talamo (via M), mentre quelle P proiettano agli strati parvicellulari (via P) [10]. Ciascuna delle vie viene utilizzata per l'analisi di aspetti differenti della stessa immagine visiva (come forma e colore) e queste informazioni vengono integrate nella corteccia per creare una rappresentazione dell'immagine nel cervello coerente con quella percepita, attraverso un insieme di regole intrinseche nel sistema nervoso centrale. Una semplice rappresentazione di quanto appena descritto è visibile nella figura 2.2.
Un'altra modalità percettiva, insieme a quelle indicate sopra, è la somatica, che include il tatto, la temperatura, il dolore e la posizione degli arti.
Una delle funzioni dei sistemi sensoriali è quella del controllo motorio. Tutti i movimenti volontari, anche quelli più semplici, hanno bisogno delle informazioni provenienti da questi sistemi. Le regioni preposte al controllo del movimento facenti parte del sistema nervoso centrale hanno accesso al flusso di informazioni sensoriali che continuamente le raggiungono. Il movimento è determinato quindi dall'interazione tra il sistema motorio e i sistemi sensoriali. Questi ultimi proiettano alla corteccia sensoriale, nel modo descritto sopra, formando una rappresentazione interna del corpo e del mondo esterno al fine di aiutare e guidare il movimento. Essa inoltre fornisce una rappresentazione comune per qualsiasi altri tipo di informazione utile sia ai gangli che al cervelletto tra i quali non esiste un collegamento anatomico diretto [7] [11].

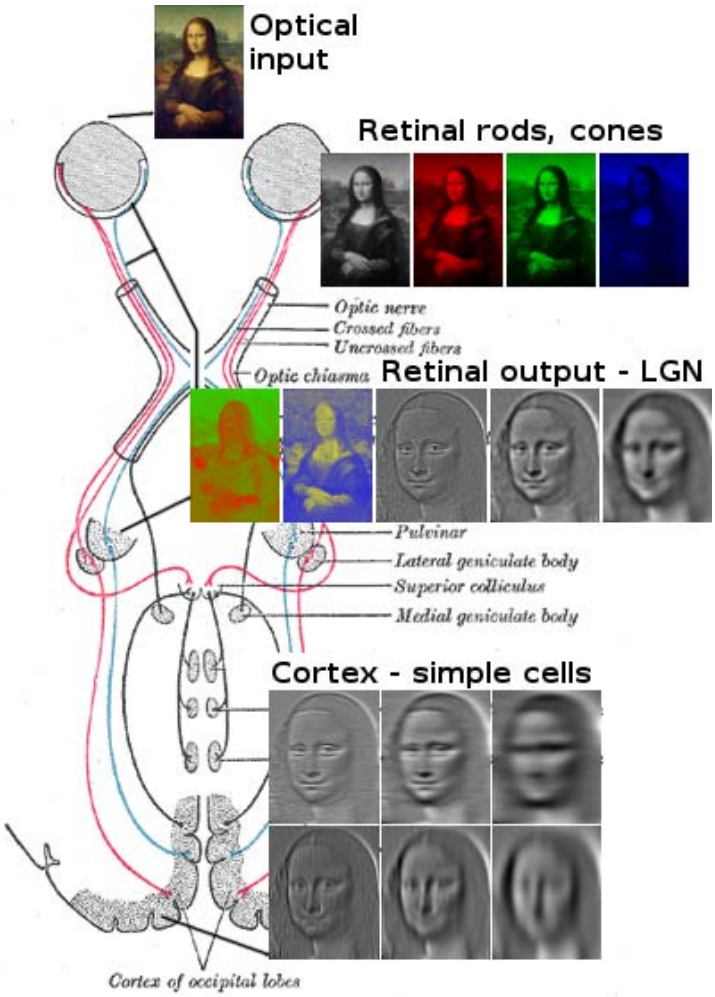

Figura 2.2: Esempio di scomposizione degli stimoli visivi da parte del cervello umano: in particolare possiamo vedere i vari livelli in cui vengono applicati i filtri; tra questi si notano i filtri riguardanti il colore e il bianco e nero e quelli inerenti alla ricerca dei lineamenti.E' possibile anche avere una visione d'insieme sulle vie che vengono utilizzate nella trasmissione delle informazioni.

Come accade per i sistemi sensoriali, la maggior parte delle aree motorie della corteccia cerebrale è organizzata in maniera somatotopica[1]; infatti, i movimenti di parti adiacenti del corpo sono controllate da aree contigue del sistema nervoso centrale [10].

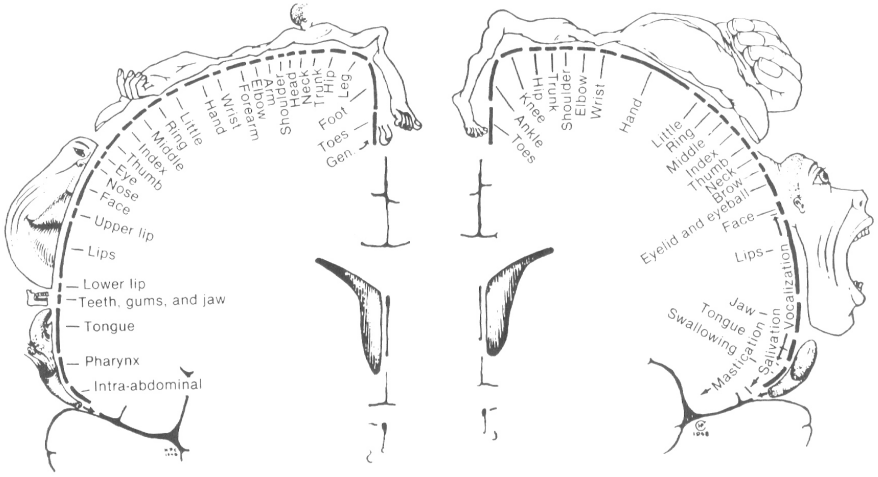

Figura 2.3: L'homunculus somatosensoriale rappresenta l'organizzazione somatotopica delle afferenze somatosensitive del corpo. Alcune regioni del copro (per esempio mano e bocca) sono visivamente piu' grandi di altre a causa di una maggiore presenza di recettori cutanei in esse presenti.

2.1.2 La generazione del movimento

Una caratteristica fondamentale degli esseri intelligenti è la capacità di muoversi. Il movimento permette di interagire con l'ambiente circonstante, di raggiungere un proprio obiettivo e svilupparne dei nuovi esplorando il mondo esterno e facendo esperienza di nuovi stimoli. Le abilità cognitive dell'uomo iniziano a svilupparsi nei primi anni di vita attraverso un confronto attivo con il mondo materiale, sociale e spaziale. Esse si riferiscono aIlea capacità di riconoscere oggetti e situazioni, ai processi di memorizzazione, del linguaggio e dell'attenzione. I traguardi raggiunti a livello cognitivo hanno una notevole influenza sullo sviluppo motorio e viceversa: i due processi sono interdipendenti. Queste idee sono riassunte nel concetto di embodi-

[1]Proprietà del funzionamento del sistema nervoso per cui ad aree adiacenti dello spazio recettivo nel mondo esterno corrispondono gruppi di neuroni adiacenti in un nucleo cerebrale o in un'area corticale.

ment [12]. Il movimento è un mezzo per conoscere, rendendo la percezione attiva, e la conoscenza permette un potenziamento delle capacità, anche di quelle motorie. Il movimento risulta un requisito necessario per gli esseri intelligenti e il suo studio ha un ruolo fondamentale per lo sviluppo di robot autonomi il cui fine è una interazione naturale ed intelligente con l'uomo.
Il sistema motorio umano genera un'ampia gamma di movimenti partendo dai riflessi fino ai movimenti volontari [2]. Possiamo distinguere due tipi di riflessi: i riflessi da stiramento e i riflessi primitivi. I primi si presentano giá dentro l'utero materno e sono delle semplici contrazioni muscolari a seguito di un estensione improvvisa mentre gli ultimi sono presenti dalla nascita e possono essere movimenti più complessi, come per esempio il riflesso di chiudere la mano quando viene toccato il palmo oppure il riflesso di Moro, che si manifesta come una apertura immediata di tutti e quattro gli arti e delle mani quando si appoggia il neonato in modo un pò brusco o come reazione ad un rumore forte.

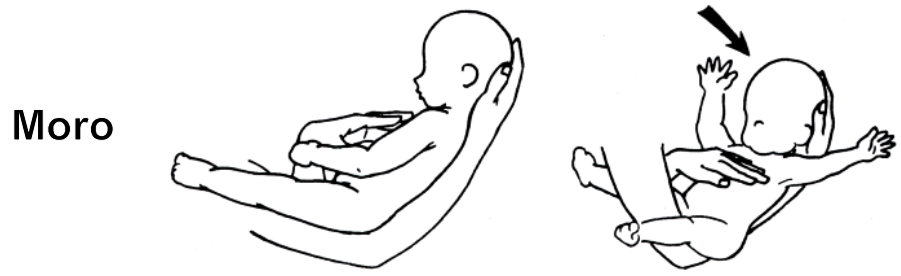

Figura 2.4: Riflesso di Moro. Ernest Moro fù il primo a descriverlo nel 1918.

Si è osservato che entrambi i tipi di riflessi sono influenzati dai centri motori sopra-spinali.
Un'altra tipologia di movimento è quella dei movimenti coordinati che non sono generati in risposta di un input né sono necessariamente acquisiti. Essa è chiamata coordinazione ereditata e la possiamo osservare ad esempio in alcuni mammiferi come i piccoli di gnu che subito dopo il parto iniziano a camminare. Nell'uomo non si manifestano episodi di coordinazione ereditata alla nascita però la presenza di questi in altri mammiferi porta a ipotizzare che anche l'uomo possieda dei comportamenti innati.
La graduale comparsa di movimenti volontari complessi che sostituiscono la maggior parte dei riflessi presenti nei primi anni di vita viene studiata principalmente da tre teorie: la mielinizzazione delle fibre nervose, che sostiene che lo sviluppo motorio dipenda dalla crescente capacità di comunicazione

tra i neuroni del sistema nervoso centrale; lo sviluppo biomeccanico, che si concentra sulla crescita delle funzionalità meccaniche del corpo umano; la parallela maturazione di differenti aree del sistema motorio. L'ultima è la teoria più accreditata e afferma che i primi comportamenti motori siano basati su un insieme rudimentale di primitive motorie innate. Altre primitive possono poi essere apprese con l'esperienza. Questi comportamenti stereotipati vengono attivati da input sia periferici che centrali. La forte presenza di riflessi nel bambino è spiegata da una scarsa maturazione dei centri motori centrali che porta ad una attivazione delle primitive motorie in risposta alla sola attivazione dei neuroni del midollo spinale ad opera di interazioni periferiche. Durante lo sviluppo del bambino, il tronco encefalico e la corteccia sensomotoria modulano in maniera sempre crescente l'attività del midollo spinale tramite l'aggiunta di assoni terminali cortico-spinali. Questi, insieme alla formazione di altre aree del sistema motorio, permettono un maggior controllo dei movimenti, la generazione di movimenti più complessi e comportamenti volontari. I riflessi primitivi non vengono persi durante lo sviluppo come ipotizzato in altre teorie ma la loro presenza viene inibita da comandi provenienti dal cervello.

Questi pattern motori che sono disponibili dalla nascita rappresentano delle prime funzioni esploratorie e di sopravvivenza, che permettono all'uomo di costruire delle mappe sensomotorie che usiamo come elementi base per altre attivitá [3]. Le attività più complesse vengono quindi generate come una combinazione di primitive motorie innate o apprese. Risulta quindi di grande importanza modellizzare queste primitive al fine di creare un sistema che possa generare dei movimenti secondo un approccio biologicamente ispirato. Esse sono anche chiamate sinergie muscolari e sono definite come una attivazione coordinata di un gruppo di muscoli nel tempo. Alcuni recenti studi supportano l'ipotesi descritta sopra e affermano che il sistema nervoso centrale le combina in base al compito che si vuole portare a termine [4]. Una prova della costruzione modulare del movimento si osserva dalla registrazione dei segnali EMG effettuata su un campione di individui eterogeneo che svolgono diverse attività.

Applicando a questi segnali degli algoritmi di decomposizione, molto usati sono l'Analisi delle Componenti Indipendenti [13] e Non-negative Matrix Factorization [14], si sono osservate delle regolarità spazio-termporali comuni tra gli individui e in parte condivise tra le diverse attività. La figura a pagina successiva illustra questa procedura.

Un'altra prova dell'esistenza di queste sinergie ci viene data dalla osservazione diretta dell'attività elettrica dell'encefalo: la scarica di un singolo neurone rappresenta l'attivazione di una sinergia, come l'attivazione di un

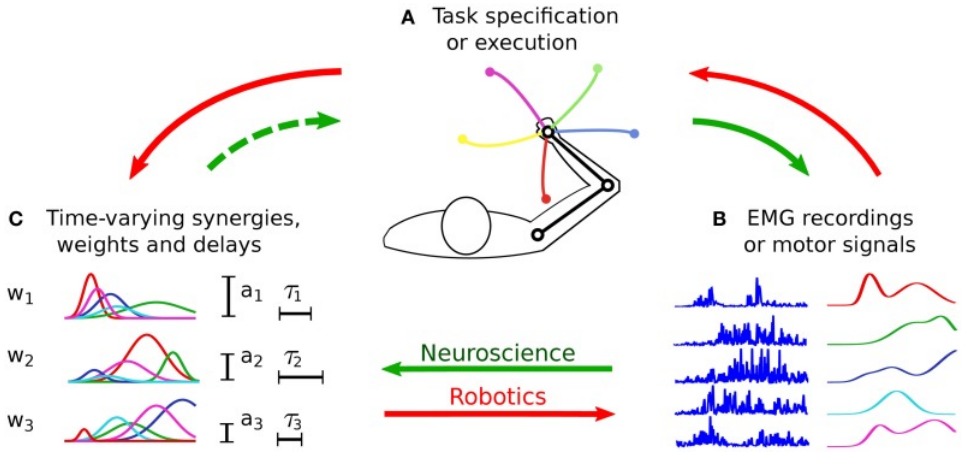

Figura 2.5: Procedura per identificare e testare le synergies. Inizialmente un gruppo di soggetti eseguono un dato compito (A). I segnali EMG acquisiti durante l'esperimento (B) sono poi analizzati, e un algoritmo di riduzione di dimensionalità viene applicato per ottenere le synergies (C). Molto spesso le synergies non sono valutate rispetto al task che si sta eseguendo (freccia tratteggiata), quindi non vi è garanzia che possano riprodurre le performance del task dove sono state osservate. In robotica (freccia rossa), le synergies sono individuate basandosi sui requisiti della classe delle attività che il robot deve svolgere (A). Così possono essere combinate per generare segnali motori (B) per eseguire al meglio lo specifico task. La qualità delle primitive è infine valutata osservando le performance tipiche del task (A).

interneurone nel midollo spinale è correlato all'attivazione di più muscoli e non uno solo. Applicando inoltre gli algoritmi di decomposizione ai segnali EMG e all'andamento delle variabili caratteristiche dell'attività che si sta eseguendo, come per esempio la distanza tra l'end effector e un oggetto nei compiti di reaching, si osserva che le sinergie implementano una specifica funzionalità biomeccanica, ovvero sono memorizzate in base al compito che svolgono. Si parla cosí di sinergie funzionali. Sebbene le attivazioni muscolari sono input di controllo del sistema muscoloscheletrico, quindi sono dei generatori nello spazio degli input, esse sono memorizzate e combinate nello spazio dei task.
Nella composizione del movimento, poi, l'esperienza sensoriale modula i parametri che sono correlati all'intensità e alla durata di ogni primitiva. L'organizzazione modulare è molto interessante perchè causa una riduzione di dimensionalità che semplifica il controllo e l'apprendimento.

2.1.3 Modelli per rappresentare le primitive motorie

In letteratura sono stati proposti diversi modelli per rappresentare le primitive motorie come i force fields (FF) [6] e le dynamic movement primitives (DMP) [5]. Un force field è una unità funzionale del midollo spinale che genera un output motorio producendo una sinergia. Le conseguenza di una sinergia è la produzione di una forza che dirige l'endpoint effector verso un punto di equilibrio nello spazio. A secondo della posizione del corpo, l'esecuzione di una sinergia produrrà una forza differente. Il force field è quindi un campo vettoriale che rappresenta per ogni posizione del corpo una forza associata alla synergy. I force fields seguono un principio di somma vettoriale, secondo il quale un force field corrispondente ad un movimento complesso che include più di una sinergy è risultato della somma di differenti force fields.

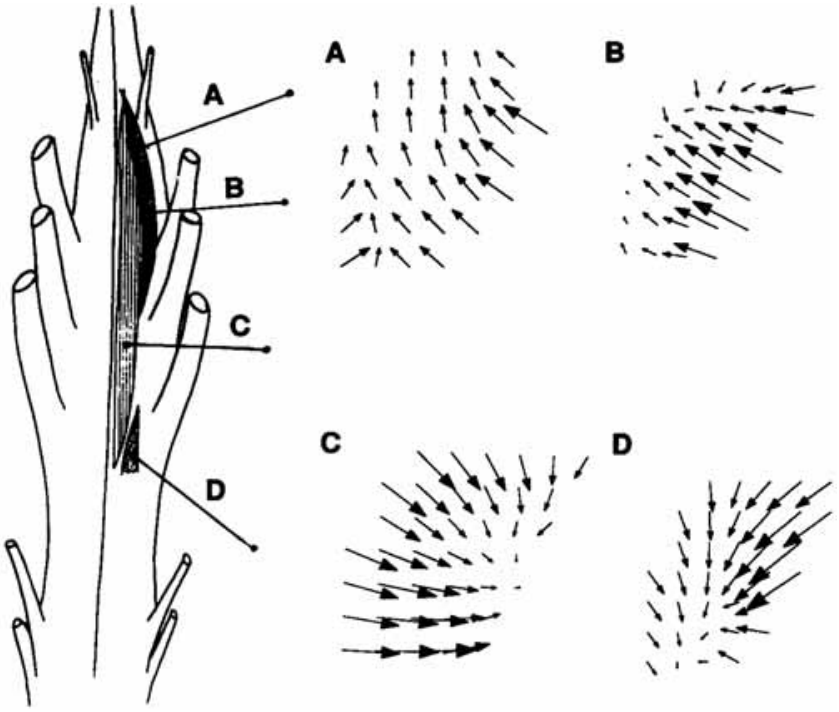

Figura 2.6: Regione del midollo spinale contenente i circuiti neurali che specificano quattro force fields (A-D).

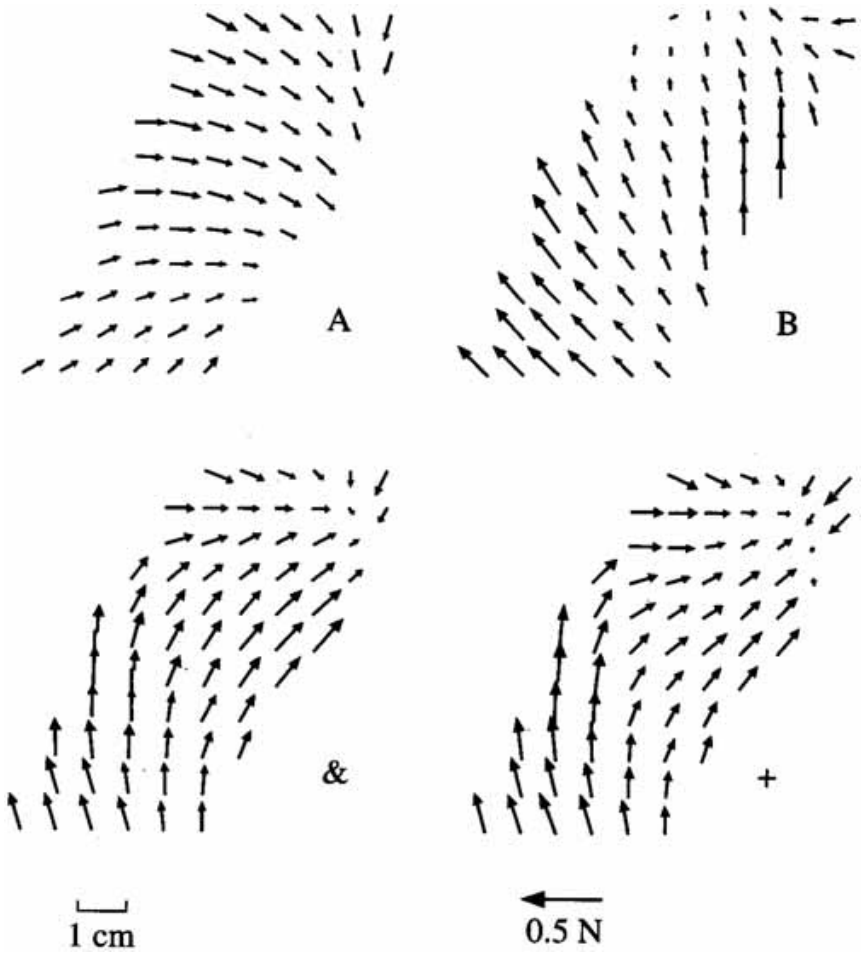

Figura 2.7: Force fields sommati vettorialmente. I campi A e B sono ottenuti in risposta alla stimolazione di due differenti siti del midollo. Il campo risultante & è ottenuto stimolando simultaneamente i due siti. Esso risulta molto simile (coefficiente di correlazione > 0.9) al campo $+$, che è stato ottenuto sommando i campi A e B.

Questi moduli possono essere visti come un alfabeto elementare dal quale, tramite la sovrapposizione, può essere generato un grande numero di azioni. La pianificazione ha luogo in termini di movimento degli effectors nello spazio. Vi è quindi un problema di trasformazione dallo spazio degli effettori a quello dei muscoli e si pensa che venga risolto trovando le sinergie che meglio approssimano il force field finale.
Un diverso approccio riconduce il problema di generazione dei movimenti ad un problema di controllo ottimo su movimenti arbitrariamente complessi senza adottare il concetto di primitiva [15]. Applicando le tecniche di programmazione dinamica si ottengono delle equazioni non lineari alla derivate parziali la cui soluzione è difficile da trovare in casi che soffrono di alta dimensionalità, come quello della generazione del movimento in sistemi complessi. La soluzione affronta il problema riscrivendo in maniera opportuna queste equazioni e introducendo delle semplificazioni. Ottiene così un sistema di equazioni differenziali ordinarie che risulta trattabile, la complessità della cui soluzione cresce linearmente con il numero degli stati. Un problema che sorge con questo tipo di approccio è la necessità di ricalcolare la soluzione per ogni nuova condizione iniziale e finale. Inoltre, sebbene suggerisca un legame tra il controllo ottimo e l'inferenza probabilistica che crea un parallelo tra la gestione delle informazioni nel sistema sensoriale e quello motorio, non sfrutta appieno le prove sull'organizzazione modulare nella generazione del movimento. Mette anzi in discussione queste ipotesi e riconduce la loro esistenza a dei segni di riduzione di dimensionalità che ogni sistema ridondante esibisce, indipendemente da come il controllore ottimo è implementato. Le difficoltà computazionali e la esigua letteratura su questa visione al momento della scrittura di questa tesi ci portano a non perseguire il modello.
Le DMP, sono invece un modello delle primitive motorie basato su un insieme di equazioni differenziali che codificano i movimenti da un punto di vista cinematico e godono di proprietà molto utili. Sono una rappresentazione che ha legami biologici molto forti, che ci permette di esprimire facilmente movimenti anche complessi. Per questi motivi abbiamo scelto di implementare le DMP nella nostra architettura. Tutta la prossima sezione sarà dedicata all'approfondimento di questo modello e delle ragioni della nostra scelta.
Le osservazioni sulla generazione del movimento evidenziate in questo paragrafo sono di ispirazione a questo lavoro tesi il cui scopo è implementare le migliori soluzioni biosipirate.

2.1.4 Teorie sulla popolazione dei neuroni

Il cervello umano è in grado di risolvere problemi computazionalmente differenti grazie ad un grande numero di aree funzionali. Anche se questo numero elevato lascia pensare che sia presente un meccanismo indipendente per ogni problema, si è visto che alla base di tutte queste aree, tra cui quelle che sono oggetto del presente lavoro di tesi, si trovano dei meccanismi comuni. Questi meccanismi si sviluppano a partire da una parte computazionale comune che sottostà ogni area funzionale: la rete neurale. Due di questi meccanismi sono la normalizzazione e il population coding [16] [17]. Il primo consiste nel mediare la somma delle risposte delle attività della popolazione di neuroni allo scopo di codificare una rappresentazione neurale. L'idea della normalizzazione nasce per giustificare la risposta fisiologica dei neuroni nella corteccia primaria visiva e, in seguito, in altre aree del cervello; questo ha portato a sostenere che la normalizzazione sia comune nelle reti neurali [18]. Viene utilizzato anche al di fuori del cervello, come nella retina. Il population coding, invece, consiste nel codificare le informazioni sensoriali attraverso un gruppo di neuroni. Ogni neurone risponde a determinati stimoli e la risposta del gruppo osservato può essere combinata per trovare nuove informazioni riguardo a quel segnale iniziale. Questi neuroni, infatti, sono spesso organizzati in strati o regioni così che neuroni vicini abbiano risposte simili [19]. Il population coding interviene per esempio nella codifica dei movimenti, nel processo decisionale [20], nel movimento oculare per focalizzare lo sguardo (movimento saccadico) e nel riconoscimento di schemi visivi [21]. Esso ha vantaggi importarti rispetto ad altre codifiche come la riduzione dell'incertezza dovuta alla risposta neurale e l'abilità di rappresentare diverse caratteristiche del segnale sensoriale contemporaneamente. E' alla base della corteccia celebrale e dei gangli, due aree del sistema nervoso di nostro interesse.

2.2 Le DMP

Un movimento può essere rappresentato con una funzione che mappa un vettore corrispondente allo stato dell'ambiente e del sistema, possibilmente tenendo conto della variabile tempo, su un altro vettore continuo che rappresenta il comando per i giunti [22]. Questa funzione dipenderà da alcuni parametri, specifici della attività che vogliamo svolgere, che dobbiamo apprendere per eseguire al meglio il movimento, come ad esempio possono essere i pesi di una rete neurale o di una generica funzione approssimante. Dato un criterio di costo che ci permette di valutare la qualità del comando

in un particolare stato, esistono diversi algoritmi di reinforcement learning per calcolare i parametri della funzione definita sopra. La modellizzazione di un movimento tipicamente risulta in un problema di controllo non lineare e con alta dimensionalità dello spazio stato-azione. L'apprendimento di questo tipo di policy risulta intrattabile anche per spazi degli stati e azioni non troppo grandi. Per questo motivo, un approccio tipico consiste nel riscrivere la policy come una combinazione di basis functions per ridurre la complessità [23]. Volendo prendere ispirazione dal funzionamento umano è naturale scegliere come basis functions delle policy più semplici come le primitive motorie descritte nei paragrafi precedenti.
La formalizzazione delle primitive adottata in questo lavoro di tesi che troviamo in [24] propone una formulazione cinematica delle policies, sostituendo come uscita del sistema i comandi motori con un cambiamento di stato, ovvero la velocità e l'accelerazione dei giunti nel tempo. Questa formulazione è più adatta al controllo motorio perchè ragionare in termini cinematici generalizza rispetto a diverse situazioni, le nonlinearità dovute alla gravità e alle forze di inerzia sono risolte dal controllore nella fase di esecuzione del movimento [25].

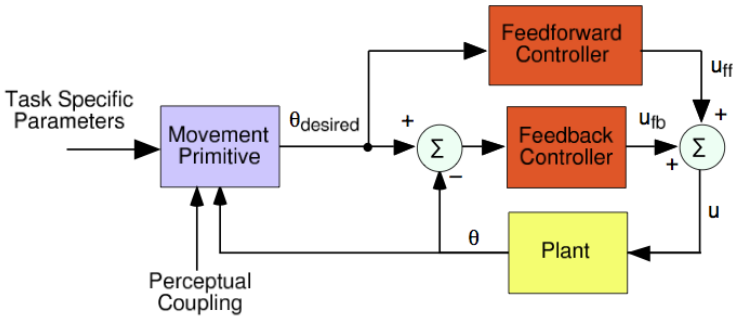

Figura 2.8: Esempio di diagramma di controllo con le dynamic movement primitives.

Le policy cinematiche possono poi essere combinate più facilmente, anche semplicemente sovrapponendole per formare comportamenti arbitrariamente complessi. Questo è uno dei motivi principali per cui si è preferito questo modello. Inoltre questo approccio è largamente ispirato da dati che mostrano come una rappresentazione di questo tipo sia adottata nella corteccia parietale [26] e dal principio dell'equivalenza motoria in psicologia [27], che mostra come insiemi di articolazioni differenti (come le dita, le gambe o le braccia) producano tutte patterns cinematicamente simili nonostante abbiano caratteristiche dinamiche molto diverse.

Due comportamenti elementari dei sistemi dinamici non lineari sono quelli attrattivi verso un punto fisso o un ciclo limite che corrispondono rispettivamente a movimenti discreti e ritmici nell'ambito del controllo motorio. Queste due tipologie di movimento attivano aree diverse dell'encefalo, con i movimenti ritmici che coinvolgono meno risorse di quelli discreti [28]. Si cercano quindi due modelli separati per i due comportamenti. I modelli proposti si chiamano dynamic movement primitives, o DMP, e l'idea è quella di sfruttare le ben note formulazioni degli attrattori descritti sopra. Una DMP consiste in due insiemi di equazioni differenziali: un sistema canonico e un sistema di trasformazione. Il primo genera un variabile di fase, che sostituisce il tempo, e viene utilizzata per rendere stabile il secondo sistema che è composto da un classico sistema lineare con un attrattore globale arricchito da un termine forzante che introduce una nonlinearità. Data una opportuna scelta dei parametri di questi sistemi e del termine forzante le proprietà di stabilità delle DMP sono garantite. Rimandiamo ai successivi capitoli i dettagli tecnici di questo modello e ci limitiamo ad aggiungere qualche informazione sulla struttura della funzione forzante. Essa è una combinazione lineare di basis function gaussiane, i pesi delle quali caratterizzano la traiettoria della DMP. Questa forma di parametrizzazione lineare è molto comoda per l'applicazione degli algoritmi di apprendimento. Un'altra importante proprietà delle DMP è l'invarianza spaziale e temporale. Cambiando i metaparametri corrispondenti alla durata del movimento, al punto attrattore o all'ampiezza dell'oscillazione, nel caso di primitiva periodica, si mantiene lo stesso comportamento qualitativo della primitiva originaria. Solo i pesi delle gaussiane determinano il comportamento qualitativo. E' una proprietà fondamentale se si vogliono riusare i movimenti adattandoli a nuove situazioni, componendoli e modificandone i metaparametri.
Un contributo importante che questa tesi si propone di dare è proprio lo sviluppo di un sistema che permetta di comporre nuovi movimenti a partire dalle DMP e relativi algoritmi di reinforcement learning che sfruttino tutte le proprietà e le caratteristiche di questa potente rappresentazione.

2.3 Modelli di apprendimento per combinare le primitive

Quando pensiamo alla natura dell'apprendimento, la prima idea che ci viene in mente è che impariamo attraverso l'interazione con l'ambiente circostante. Quando un bambino gioca, afferra degli oggetti, muove le braccia, stabilisce autonomamente una connessione sensomotoria con la realtà che lo circonda.

Esercitando questa connessione si producono informazioni sulle cause e gli effetti delle azioni e su cosa fare per soddisfare dei propri obiettivi. Imparare tramite l'interazione con l'ambiente è una idea fondamentale che sta alla base di quasi tutte le teorie dell'apprendimento e dell'intelligenza.

2.3.1 Apprendimento per rinforzo, supervisionato e non supervisionato

L'approccio maggiormente esplorato in questo lavoro di tesi è chiamato apprendimento per rinforzo, o reinforcement learning [23]. Esso consiste nell'imparare cosa fare, quindi come far corrispondere azioni a specifiche situazioni, allo scopo di massimizzare un segnale numerico di rinforzo. L'agente, che può essere un animale, un uomo o un sistema artificiale, deve scoprire quale azione porta il rinforzo maggiore provandole, interagendo con l'ambiente.
Due caratteristiche fondamentali che distinguono il reinforcement learning da altri sistemi di appredimento sono la ricerca attraverso un processo di trial-and-error e la possibilità di avere rinforzi ritardati nel tempo, ovvero quando l'esecuzione di una azione non influenza solo una ricompensa immediata ma anche quelle derivanti dalle situazioni e azioni successive. L'apprendimento per rinforzo è definito specificando il problema di apprendimento. Esso deve catturare tutti gli aspetti più importanti del problema reale che l'agente affronta. Abbiamo già incontrato il primo elemento che caratterizza un problema di reinforcement learning, l'agente. Ciò con cui esso interagisce è l'ambiente, il secondo elemento del problema. Si definisce ambiente tutto ciò che non può essere arbitrariamente cambiato dall'agente. Gli altri elementi sono gli stati, cioè rappresentazioni dell'ambiente in un certo istante, il rinforzo e le azioni, che possono essere anche completamente mentali.

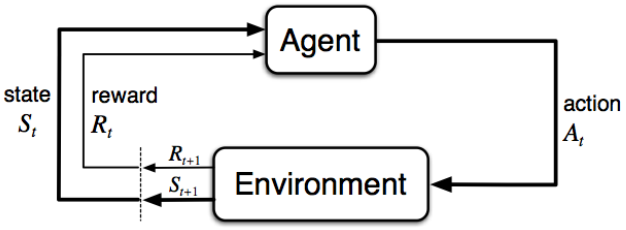

Figura 2.9: Interazione agente-ambiente nel reinforcement learning.

Ad ogni istante di tempo l'agente implementa una funzione definita dagli

stati alle probabilità di selezionare ogni possibile azione. Questa funzione è chiamata policy dell'agente ed è costruita selezionando una azione ad ogni istante di tempo e osservando il rinforzo corrispondente e lo stato all'istante successivo. Un metodo di reinforcement learning specifica come l'agente cambia la propria policy, non il problema di apprendimento. Esistono diversi metodi per affrontare questo tipo di problemi e nel prossimo paragrafo verrà introdotto il metodo chiamato Actor-Critic, molto legato a modelli psicologici e biologici [23].
Altri due approcci di apprendimento vengono integrati in questa tesi, quello supervisionato e non supervisionato. Il primo consiste nell'imparare a partire da esempi forniti da un supervisore esterno. In questo modo il sistema impara a produrre i risultati attesi a partire dai dati in ingresso. E' un approccio molto usato ma da solo non è adeguato per l'apprendimento tramite l'interazione. L'apprendimento non supervisionato invece prevede il passaggio di dati al sistema senza fornire ulteriori informazioni come uscite previste o ricompense in base alle azioni svolte. Lo scopo è di riuscire a trovare degli schemi tra i dati e riuscire a raggruppare i differenti gruppi focalizzandosi sulla classificazione. Un'architettura che implementa assieme diversi approcci si dice mixed model, basato cioè sull'utilizzo di differenti modelli cooperanti, ed è molto ben conosciuto dalla comunità neuroscientifica, anche se la sua applicazione nel controllo dei robot non è ancora molto diffusa [29].

2.3.2 Metodi Actor-Critic

I metodi Actor-Critic rientrano nella classe dei metodi di Temporal Difference, o TD, per la soluzione del problema di controllo di reinforcement learning [30] [31]. L'idea alla base dell'apprendimento TD è quella di imparare direttamente dall'esperienza, senza la necessità di un modello della dinamica dell'ambiente, e di riuscire a migliorare la valutazione delle azioni da compiere basandosi su altre valutazioni ricavate in momenti precedenti. La peculiarità dei metodi Actor-Critic è quella di avere una struttura di memoria separata per rappresentare esplicitamente la policy, indipendente dalla funzione di valutazione degli stati, elemento utile agli algoritmi di RL che indica quanto è buono per l'agente trovarsi in dato stato.
La struttura rappresentante la policy è chiamata Actor, perchè è usata per selezionare le azioni, e la funzione di valutazione degli stati, o meglio delle coppie stato-azione, è chiamata Critic perchè valuta le azioni decise dall'attore. L'apprendimento viene definito on-policy: il critico deve valutare qualsiasi politica l'attore stia seguendo e non è possibile con questi meto-

di generare esperienza seguendo una policy e impararne un'altra. Il critico prende la forma del segnale tipico di temporal difference che guida l'apprendimento sia dell'attore che dello stesso critico.

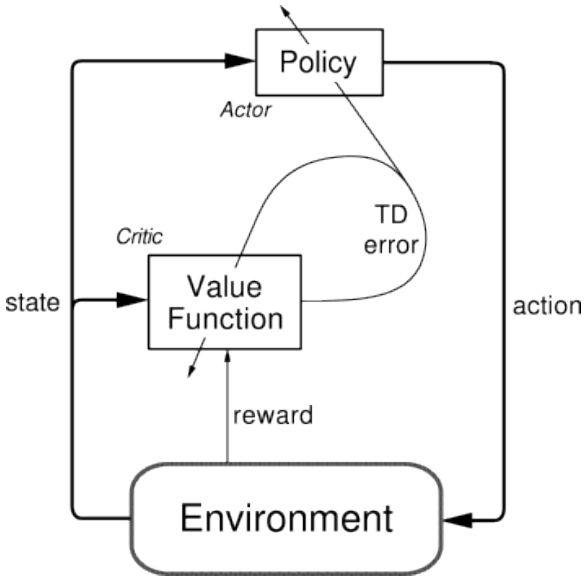

Figura 2.10: Modello Actor-Critic

Esistono diverse opzioni per implementare questa architettura. Se lo spazio delle azioni è piccolo, il critico può usare una funzione approssimante, o assumere una struttura tabulare se si ha uno spazio degli stati discreto, e l'attore seguire una strategia di esplorazione ϵ-greedy o di Boltzmann [30]. Se invece lo spazio delle azioni è grande e continuo, anche l'attore può usare la stessa tecnica del critico. In letteratura si trovano molti algoritmi che utilizzano questa architettura e permettono di risolvere il problema del controllo [32] [33] [34] [30]. Ogni algoritmo è indicato per una specifica classe di problemi.
Nel capitolo 4 verranno illustrati gli algoritmi sviluppati e implementati per affrontare il problema della generazione del movimento che tengano il più possibile in considerazione le caratteristiche biologiche descritte in questo e nei successivi capitoli.

Capitolo 3

Motivazioni di un modello bioispirato

In questo capitolo mostriamo l'idea dalla quale ha preso spunto questa ricerca. Viene descritto poi l'approccio utilizzando alla risoluzione del problema affrontato e vengono introdotti i vari moduli che compongono il modello implementato.

3.1 Il problema della generazione del movimento

Uno degli scopi principali della robotica è quello di creare agenti artificiali in grado di coesistere e interagire con l'uomo nella società. Negli ultimi anni sono stati sviluppati robot antropomorfi e umanoidi che hanno posto importanti sfide in questo campo. Sono molto complessi, con un alto numero di gradi di libertà e le tecniche usate in passato per il loro controllo non sono più applicabili. Si cercano nuovi metodi per controllare l'esecuzione del movimento che tengano conto anche di altre complessità che conseguono dalla vastità del campo di applicazione di questi robot. Integrarsi nella società significa far fronte alle situazioni che gli uomini affrontano ogni giorno, a partire dai frequenti contatti con un ambiente di cui non si conoscono con esattezza nel dinamiche. Inoltre la pianificazione del movimento in sistemi motori ad altà dimensionalità pone un'altra difficile sfida: se una efficiente costruzione del movimento in robot industriali, tipicamente dotati da tre a sei gradi di libertà, è un problema complesso, ottenere un controllo ottimo, per di più in un ottica di controllo in real-time, su sistemi da 30 o 40 gradi di libertà è particolarmente scoraggiante.
Un approccio molto usato in robotica è quello del reinforcement learning.

Esso offre un framework e un insieme di strumenti per la costruzione di comportamenti difficili da gestire con altri approcci. Anche questo strumento però non permette di superare da solo le complessità descritte prima. Difficilmente si riesce a risolvere un problema di apprendimento oltre le sei dimensioni dello spazio continuo degli stati e delle azioni. Di solito questi spazi sono molto grandi perchè aumentano esponenzialmente nel numero di variabili di stato e queste sono spesso per di più continue. Questo problema è anche conosciuto come "the curse of dimensionality". I nuovi robot poi integrano molteplici modalità sensoriali oltre la visione, come i sensori tattili, quelli acustici, olfattivi o reti di sensori distribuiti in genere. Oltre a trovare metodi affidabili per elaborare questa ricca percezione dell'ambiente esterno, incorporare queste informazioni nel sistema di generazione dei movimenti aumenta il problema di pianificazione e controllo.
La situazione che ci si trova di fronte si avvicina molto a quella che la biologia affronta con successo ogni giorno. Una tendenza degli ultimi anni è quella di studiare i comportamenti neurobiologici e cognitivi per estrarre principi rilevanti al fine di costruire nuovi metodi per la robotica avanzata. Si vuole replicare la flessibilità, la plasticità e la velocità con la quale il cervello umano risolve il problema del movimento prendendo ispirazione dai suoi meccanismi.
Lo sviluppo cognitivo è il processo di miglioramento delle capacità mentali che interviene per tutta la durata della vita di un uomo. Si concentra su come una persona percepisce, pensa e prende coscienza del proprio mondo attraverso l'interazione con esso. Una caratteristica fondamentale è la capacità di sviluppare nuovi obiettivi autonomamente e di conseguenza nuovi comportamenti che permettono di soddisfare i propri obiettivi. Non è questo l'unica ragione dell'insorgenza del movimento, come descritto nel capitolo precedente, ma è la ragione chiave della presenza dei movimenti detti volontari. La tesi precedentemente proposta riguardante questi aspetti si è concentrata sulla costruzione di un modello robotico bioispirato che permette ad un agente di sviluppare autonomamente nuovi obiettivi rispetto a quelli decisi a priori [8]. Essa costituisce la parte cognitiva che va a integrarsi con l'architettura proposta in questo lavoro per creare un sistema di generazione di movimenti volontari per un robot che può definirsi autonomo. Il livello cognitivo è costituito dall'architettura IDRA che ricrea il funzionamento delle aree del cervello che svolgono un ruolo fondamentale nello sviluppo cognitivo: amigdala, talamo e corteccia cerebrale [1]. E' formata da una rete di componenti elementari chiamati Moduli Intenzionali e ognuno di questi contiene al suo interno altri più specifici che modellizzano la corteccia (che crea una rappresentazione del mondo esterno), il talamo

(il fulcro dello sviluppo di nuovi obiettivi) e l'amigdala (che si attiva per segnalare istinti e paure innate della persona).
Analogamente anche il problema del movimento viene risolvo dal cervello attivando in contemporanea molte aree. Si vuole dare ora una visione d'insieme dei circuiti che si creano nelle tre fasi che possiamo individuare nella generazione del movimento:

Codifica degli stimoli Le prime aree del cervello attivate sono quelle che codificano gli stimoli sensoriali. Come già descritto nel capitolo precedente, esistono molte prove scientifiche che suggeriscono che diverse modalità sensoriali vengono interpretate da specifiche aree funzionali della corteccia celebrale e che le informazioni percepite vengono filtrate dagli organi di senso prima di arrivare al cervello. Le più importanti aree coinvolte in questa fase appartengono quindi alla corteccia sensoriale e sono la corteccia primaria visiva, la corteccia somatosensoriale primaria e la corteccia parietale posteriore. La corteccia primaria visiva, o V1, localizzata nel polo posteriore del lobo occipitale, è la prima area dove si uniscono i segnali provenienti dai due occhi per codificare una informazione binoculare, utile per capire la profondità. Svolge anche le funzioni di riconoscimento di oggetti e di movimenti [35].

Figura 3.1: Corteccia Primaria Visiva

Si pensa che adotti il meccanismo di statistical coding, che è una forma di population coding impiegato per i dati sensoriali. E' un meccanismo comune a tutte le aree del cervello che svolgono una pre-elaborazione dei dati, come la corteccia somatosensoriale, localizzata nel lobo parietale, che tra le

sue funzioni vanta la ricezione di stimoli sensitivi del tatto e delle informazioni propriocettive. Il principale vantaggio di adottare questa codifica è la riduzione della dimensionalità dello spazio degli inputs [36]. La corteccia parietale posteriore, o PPC, riceve in input le attivazioni neurali dalle principali aree sensoriali e integra le informazioni per creare un stato complesso su cui altre aree, come quelle motorie, possono lavorare. Svolge molte funzioni come quella di localizzazione visuospaziale ed interviene nei processi dell'attenzione e del linguaggio.

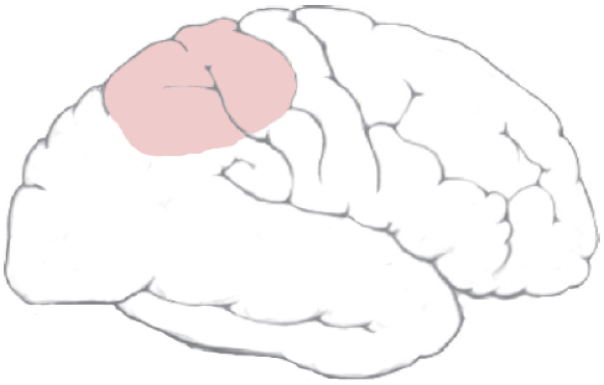

Figura 3.2: Corteccia Parietale Posteriore

Calcolo del movimento Possiamo distinguere due tipologie di movimenti volontari: quelli guidati da uno stimolo esterno e quelli generati internamente. L'area corrispondente al cervelletto svolge un ruolo chiave per la prima tipologia. Esso è collocato nella fossa cranica posteriore e presenta diversi canali paralleli che, passando attraverso il talamo, arrivano alla corteccia, non solo motoria, ma anche prefrontale, temporale e parietale [11]. La diversità di queste aree corticali è in accordo con il suo coinvolgimento in diversi processi come quelli dell'attenzione, della memorizzazione e del linguaggio. Si pensa abbia il compito di adattare il movimento in tempo reale corrispondentemente ad un feedback come può essere quello visivo. Il calcolo più critico in questo caso corrisponde ad una trasformazione di coordinate, da quelle visive a quelle motorie. Recenti studi hanno affermato che ha una struttura anatomica che gli permette di apprendere secondo un approccio error-based [7]. Nei movimenti guidati dalla memoria invece la parte più critica è la selezione, o la costruzione, dell'azione più appropiata che richiede la conoscenza o la previsione di un segnale di rinforzo. I gangli

alla base permettono questo tipo di apprendimento basato sul rinforzo. Costruiscono, come il cervelletto, diversi circuiti paralleli con la corteccia, sia al fine di ricevere una codifica dell'input sensoriale come della partecipazione a processi di immaginazione, planning, attenzione. Possiamo individuare un "gradiente" di informazioni provenienti dalla corteccia, a partire da quelle sensomotorie a quelle motivazionali e associative. Questo fa pensare alla presenza di diversi processi di apprendimento in cascata, ognuno con una propria rappresentazione di stati, valori e azioni [37].

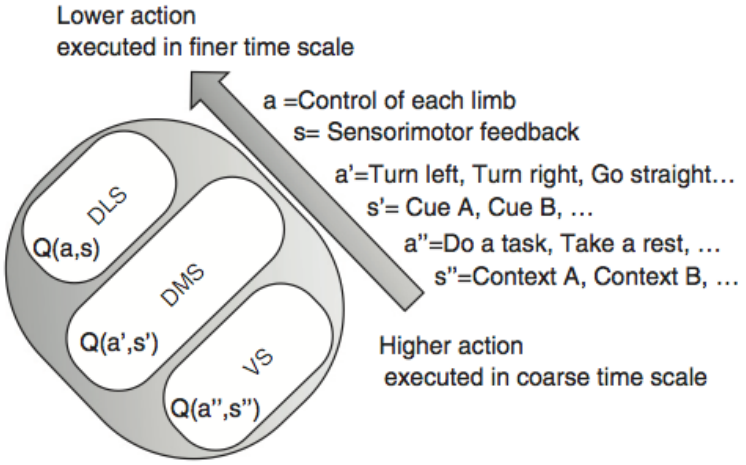

Figura 3.3: Struttra gerarchica dei gangli alla base che evidenzia moduli di apprendimento a diversi livelli di astrazione

Esecuzione del movimento Come vedremo nel dettaglio nel capitolo successivo, alcune aree del cervello possono essere classificate secondo il paradigma di apprendimento in cui sono specializzate. Abbiamo già incontrato il cervelletto, specializzato nell'apprendimento supervisionato. Esso svolge anche il compito di memorizzare un modello inverso della dinamica che ci permette, a partire da una rappresentazione cinematica del movimento in termini di posizioni , velocità o accelerazioni dei giunti, di calcolare le forze da applicare. Riceve le traiettorie costruite dalla corteccia premotoria, che ha l'importante compito di mantenere gli schemi motori acquisiti, siano essi azioni complesse o semplici attivazioni di primitive di moto.

3.2 Il modello bioispirato proposto

Questo lavoro di tesi vuole creare una modello biologicamente ispirato per la pianificazione e l'apprendimento del movimento che sia adatto al controllo di sistemi complessi e che possa integrarsi con un sistema cognitivo di sviluppo autonomo di obbiettivi. Abbiamo scelto un approccio bioispirato perchè, oltre ai motivi evidenziati sopra, esso ha il grande vantaggio di non essere task-specific. Un approccio classico alla robotica si basa su strumenti matematici per costruire delle traiettorie desiderate con la necessità di conoscere il modello della dinamica del robot. Un meccanismo che si ispira alla biologia invece permette di adattarsi a strutture differenti, sfruttando la loro morfologia senza considerare esplicitamente un modello della dinamica. Abbiamo scelto un approccio per la generazione del movimento che propone l'esistenza di primitive motorie come elementi base. Esistono diverse teorie che propongono definizioni differenti delle primitive di moto, come descritto nel capitolo due. In questo lavoro abbiamo adottato le primitive funzionali, ovvero dei movimenti stereotipati che il cervello umano memorizza in base al compito che realizzano, come per esempio quello di raggiungimento o di equilibrio. Il primo passo verso l'implementazione di un modello è stato scegliere una rappresentazione matematica delle primitive. Le Dynamic Movement Primitives sono il modello di primitive trovato in letteratura che meglio rispecchia la rappresentazione dei movimenti nel cervello. Inoltre sono anche uno strumento molto potente che permette di generalizzare il movimento rispetto a diverse situazioni e creare movimenti arbitrariamente complessi tramite una sovrapposizione di DMP. La soluzione proposta in questo lavoro di tesi prevede una combinazione di DMP più complessa di una semplice sovrapposizione. Il movimento finale viene generato prima combinando tra loro, con un funzionamento simile ad un rete di esperti, le primitive funzionali che realizzano uno stesso task e infine sovrapponendo le combinazioni corrispondenti a task diversi.
Questo modello corrisponde al funzionamento della corteccia premotoria e insieme ai modelli per la corteccia sensoriale e per i gangli alla base costituisce la soluzione proposta. L'idea è di ricreare l'organizzazione cerebrale descritta nel paragrafo precedente. Le varie aree vengono modellizzate a livelli di astrazione differenti, replicando in alcune aree alcuni meccanismi computazionali del cervello come il population coding. Il modello corrispondente ai diversi tipi di corteccia sensoriale, che codificano il segnale in ingresso al sistema, è stato ricavato dall'architettura IDRA. La motivazione è quella di condividere lo stesso sistema di codifica dell'input visto che le due architetture dovranno essere integrate. In IDRA questa funzionalità è

stata implementata in un componente chiamato modulo di categorizzazione che presenta la stessa struttura per qualsiasi area sensoriale esso stia riproducendo, esattamente come le diverse aree sensoriali nel cervello presentano simile struttura anatomica e siano potenzialmente in grado di codificare segnali di diverse modalità [38]. Il modulo di categorizzazione implementa un algoritmo di decomposizione ricreando così il meccanismo di statistical coding. Nello nostro modello vi è un modulo di categorizzazione che fornisce l'input al modulo di apprendimento per rinforzo, corrispondente ai gangli della base del cervello umano. Questa è l'area rappresentata nel maggior dettaglio perchè svolge un ruolo critico nella generazione del movimento, quello dell'apprendimento, sia di primitive base che di azioni complesse. Il nostro modello dei gangli realizza un apprendimento per rinforzo model-free, come suggerito in [37] per la generazione di una prima forma di movimenti complessi, per determinare il valore ottimale dei parametri usati nella combinazione delle primitive ad opera del modulo della corteccia motoria. Lo abbiamo descritto con una architettura Actor-Critic che somiglia molto alla struttura anatomica dei gangli nel cervello. Abbiamo scelto un algoritmo di RL basato su due scale di tempo differenti per l'apprendimento dell'Attore e del Critico. Questo tipo di algoritmo permette di avere una rappresentazione esplicita della funzione che valuta le coppie stato-azione implementata nel Critico, che ricerchiamo in questa forma sempre per avere un parallelo biologico, e generalizzare il sistema proposto sia a problemi episodici che non episodici. In particolare, l'algoritmo implementato effettua una gradiente naturale che risulta più performante rispetto ad altre soluzioni che possono essere implementate [39]. Ci siamo concentrati sulla generazione di movimenti basati sulla memoria quindi non è stato necessario implementare un modello per il cervelletto per controllare i movimenti guidati da stimoli esterni.

Il software è stato sviluppato nell'ambiente per il calcolo numerico MATLAB (Matrix Laboratory), che fornisce un ambiente dove scrivere facilmente il codice, interfacciarsi con il robot usato per gli esperimenti e produrre una rappresentazione grafica dei risultati. Abbiamo effettuato tre tipologie di test con il robot antropomorfo NAO. I test riguardano lo svolgimento di un gioco per il robot che consiste nel coprire con un bicchiere un pallina ferma su un tavolo. Sono state proposte due varianti dove il NAO impara come eseguire questo movimento a partire da un dataset di primitive motorie molto simili a dei riflessi neonatali, prima applicando solo l'algoritmo di apprendimento per la combinazione delle primitive, poi integrando il sistema con l'architettura IDRA personalizzata per il nostro sistema e per test.

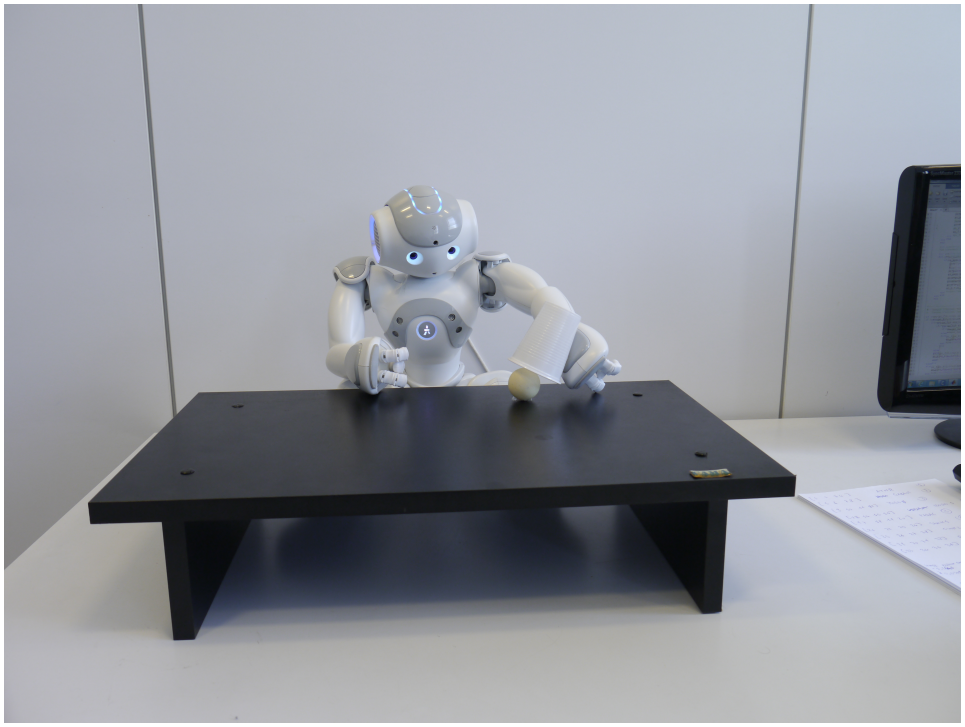

Figura 3.4: Esperimento condotto con il robot NAO

Capitolo 4

Architettura del sistema

In questo capitolo mostriamo il modello che abbiamo sviluppato e il suo funzionamento senza addentrarci nei particolari implementativi che verranno discussi nel capitolo successivo. Viene discussa la soluzione proposta a partire dalle unità atomiche del sistema, le DMP proposte in [5], per poi passare al modulo delle corteccia premotoria che le combina per generare il movimento complesso. Dopo avere introdotto i relativi algoritmi di apprendimento, si illustra l'architettura IDRA, usata per codificare lo stato e generare un segnale di interesse, con cui il modello di questo lavoro di tesi si integra. Nella figura 4.1 diamo una visione d'insieme del modello proposto.

4.1 Il movimento e le DMP

Dare una definizione di movimento è fondamentale per costruire il nostro modello. Abbiamo gi à illustrato il concetto di primitiva motoria e le motivazioni che ci portano a pensare che il movimento sia costruito a partire da questi moduli base. In particolare il concetto di primitiva funzionale è molto interessante e sembra che descriva meglio di altri modelli le prove scientifiche raccolte su questo argomento. In base a questo concetto un movimento volontario può essere una singola primitiva, oppure una azione complessa definita come combinazione di primitive motorie semplici, ognuna con un suo scopo. A sua volta una combinazione di primitive può essere una sequenza di queste, una sovrapposizione o entrambe. I moduli base poi possono essere ciclici, come quelli che compongono l'azione di camminare, oppure discreti, come per esempio un dritto nel gioco del tennis.
Nel capitolo precedente abbiamo illustrato una teoria sull'organizzazione gerarchica della costruzione dei movimenti nel cervello. Si osserva nell'area dei

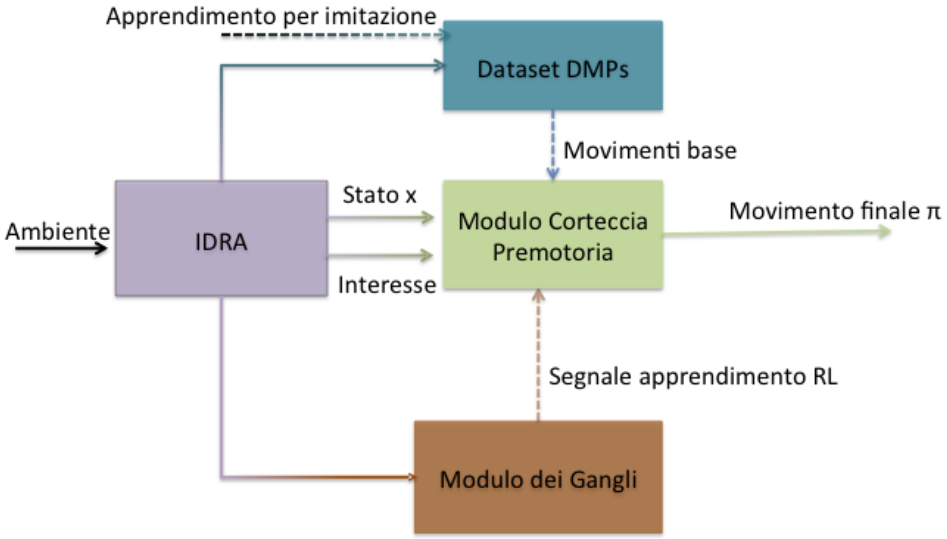

Figura 4.1: I moduli che compongono il modello proposto. Il primo componente che si attiva è IDRA che restituisce una rappresentazione dello stato attuale ai rimanenti componenti. Successivamente si attiva il modulo dei gangli che ha il compito di addestrare quello della corteccia, il quale genera il movimento finale a partire da un dataset di primitive appreso per imitazione.

gangli la presenza di moduli di apprendimento in cascata che definiscono, a scale di tempo e astrazione differenti, elementi motori base per i moduli più in alto nella gerarchia [37]. Questa teoria convalida la nostra definizione di movimento e ci suggerisce che ogni livello di astrazione debba avere un suo modello e un metodo di apprendimento. Il livello più in basso nella gerarchia è la primitiva. Abbiamo scelto le DMP per modellizzare questo elemento. Data la loro formulazione in termini cinematici, esse si prestano bene al meccanismo di generazione del movimento scelto nel nostro modello. Sono composte da due sistemi di equazioni differenziali, uno che definisce una variabile di fase, usata come sostituto del tempo, l'altro che descrive la traiettoria, relativa ad un grado di libertà del sistema. Per poter descrivere traiettorie arbitrariamente complesse, l'ultimo sistema presenta un termine forzante rappresentato da una sorta di kernel smoother la cui formulazione matematica verrà chiarita nel capitolo successivo [40]. Nonostante la presenza di un termine non lineare, i due sistemi insieme hanno un comportamento asintoticamente stabile. La particolare forma matematica della funzione forzante lineare nei parametri, ci permette di definire più facilmente gli algoritmi di apprendimento.
Individuiamo due tipi di apprendimento per le primitive, che sono l'apprendimento per imitazione, al fine di imparare nuove primitive da esempi esterni, e quello per rinforzo, per migliorare i comportamenti osservati. In questa tesi abbiamo implementato il primo tipo di apprendimento dando per scontato che le primitive acquisite fossero già efficienti per il robot. Abbiamo preferito concentraci sul fornire al nostro sistema un metodo per aquisire le primitive dall'esterno, perchè questa fase risulta fondamentale per l'acquisizione di un dataset di movimenti su cui poi progettare un algoritmo di combinazione delle primitive. Esistono già diversi algoritmi abbastanza performanti per l'ottimizzazione delle singole primitive mentre sono state proposte poche soluzioni per la costruzione di movimenti complessi [41] [42]. Per questo motivo non affrontiamo il problema dell'ottimizzazione delle primitive ma forniamo un metodo di regressione per costruire le DMP a partire da esempi di traiettorie composte dalla posizione dei giunti nel tempo. Integrando il sistema rappresentante la fase otteniamo un semplice problema di supervised learning dal quale otteniamo i pesi della funzione non lineare.

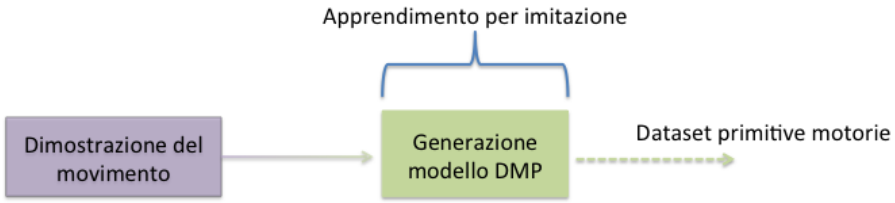

Figura 4.2: Schema a blocchi dell'apprendimento delle DMP

Il modello dipende da alcuni metaparametri come la durata, o il punto di arrivo del movimento, che possono essere variati per adattare il movimento a nuove situazioni, mentre il suo andamento qualitativo rimane inalterato essendo caratterizzato solo dai pesi definiti precedentemente. Questo fa si che i movimenti possano essere categorizzati in base a questo insieme di parametri.
Non abbiamo sviluppato degli algoritmi che sfruttino le due ultime proprietà però esse rappresentano dei motivi ulteriori che ci hanno spinto ad adottare questo modello nell'ottica di una costruzione di una architettura motoria più completa.

4.2 La combinazione di primitive

Salendo nella gerarchia che controlla la costruzione del movimento troviamo l'area della corteccia motoria preposta alla combinazione delle primitive motorie. Anche per questo livello forniamo un modello e un algoritmo di apprendimento. Il modello proposto costruisce il movimento finale in due fasi: prima si calcola una media pesata dei gruppi di primitive che controllano l'andamento temporale della stessa variabile, ovvero delle primitive funzionali che hanno lo stesso scopo (come può essere quello di reaching), e poi si sommano i contribuiti di questi gruppi di movimenti per formare quello finale. Questo modello è in linea con i recenti studi neuroscientifici e inoltre sfrutta la possibilità di sovrapposizione delle DMP. L'apprendimento del modello avviene per rinforzo. Il modulo descritto nel paragrafo successivo, corrispondente all'area dei gangli, implementa un algoritmo biosipirato per ottimizzare i parametri rappresentati dai pesi usati nella combinazione dei gruppi di primitive funzionali. In questo modo il modello della corteccia premotoria si comporta come una "mixture of experts", usa in pratica una combinazione di esperti più semplici, rappresentati dalle primitive, per generare il movimento complesso. Ogni esperto copre regioni di input differenti, vale a dire ogni movimento è più indicato per una data situazione. Bisogna però decidere quale esperto usare e quanto valutare la sua risposta. I pesi definiti sopra svolgono questo compito e rappresentano la"gating network" del modello. Questa soluzione risulta ancora maggiormente bioispirata perchè implementa il meccanismo di population coding visto in precedenza, dove i pesi rappresentano dei neuroni, ognuno con una risposta agli input, ovvero con quale probabilità attiverò la primitiva in un dato stato, che segue una data distribuzione. Le risposte di tutti i neuroni sono poi combinate.
L'apprendimento in una mixture of experts consiste di due fasi: si apprendono i parametri degli singoli esperti e la gating network. La prima fase la abbiamo già svolta per imitazione, la seconda la approfondiamo nel prossimo paragrafo.

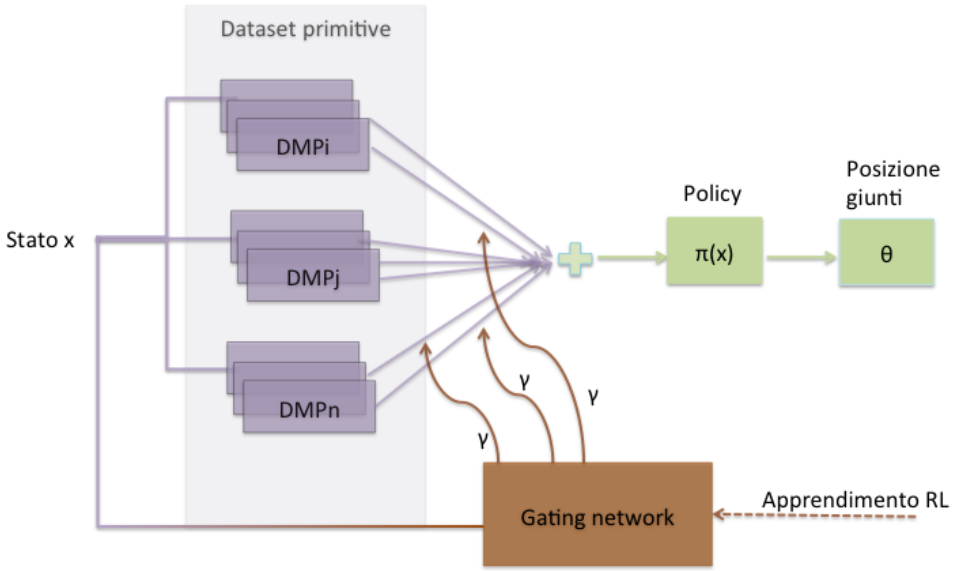

Figura 4.3: Struttura del modello della corteccia premotoria

4.3 L'apprendimento dei movimenti complessi

Il modulo dei gangli alla base riceve in input lo stato del sistema e restituisce un valore compreso tra zero ed uno che corrisponde al peso da dare all'esperto i-esimo. L'esecuzione dell'algoritmo di apprendimento avviene quindi in parallelo per ogni esperto. Il criterio che abbiamo seguito nella progettazione dell'algoritmo di apprendimento della gating network è stato quello di implementare una soluzione che risulti il più possibile collegata alla biologia. Alcuni studi scientifici hanno mostrato come il funzionamento della corteccia e dei gangli alla base possa essere descritto non tanto chiedendosi quali funzioni svolgano, che abbiamo osservato essere tante per ogni area, ma invece quale è il metodo del loro funzionamento [7]. Studiando la struttura anatomica della corteccia si è osservato che essa è specializzata nell'apprendimento non supervisionato, mentre i gangli alla base sono specializzati nell'apprendimento per rinforzo. In particolare questi ultimi presentano, all'interno di una sua area chiamata striatum, due regioni distinte. La prima forma un circuito con il sistema dopaminergico, che ha la funzione di codificare il rinforzo derivante da una situazione, la seconda invece è collegata tramite il talamo ad alcune aree della corteccia celebrale, tra cui quella motoria. Questo ci porta a pensare che vi siano aree distinte per la valutazione dello stato e per la selezione dell'azione da compiere. Per questo motivo il modello dei gangli che proponiamo ha una struttura Actor-Critic. Inoltre si è osservato che i neuroni che portano il segnale di rinforzo si attivano quando incontriamo un reward inaspettato, come se codificassero un errore nella predizione del reward. Scegliamo quindi una struttura per il Critico che si basi sull'aggiornamento di una funzione che valuti lo stato e le azioni.
La struttura del modello si basa su un Attore che ha la forma di una policy parametrica gaussiana e su un Critico, descritto con una funzione approssimante lineare nei parametri per un spazio degli stati grande, che valuta l'operato di una policy che può cambiare ad ogni istante di tempo. Per descrivere un modello che funzioni in ogni tipo di task, anche non episodico, abbiamo pensato di usare la tecnica degli algoritmi a due scale di tempo per far apprendere il modello del Critico più velocemente della policy dell'Attore così che possa valutare meglio il suo comportamento quando ancora non ha raggiunto la convergenza [30]. L'algoritmo che abbiamo implementato su questi moduli è il Natural Actor Critic (NAC) [33] con SARSA(1) [23] per il Critico e una policy gaussiana per l'Attore. Abbiamo scelto questi algoritmi perchè sono i migliori algoritmi trovati al momento della scrittura di questa tesi che rispettano il funzionamento biologico sopra evidenziato. Escludiamo

quindi tutti gli algoritmi denominati direct policy search, che usano i reward senza mantenere una funzione di valutazione delle coppie stato e azione, dei quali alcuni esempi possono essere trovati in [43] [42]. L'algoritmo NAC segue il gradiente naturale per aggiornare i parametri della policy, eseguendo il gradiente ascendente direttamente nello spazio della policy invece che nello spazio dei parametri. La scelta di SARSA(λ) invece è giustificata, oltre che dalla biologia, anche dalla garanzia di (quasi-)convergenza che aumenta con l'aumentare di λ verso il valore 1. Inoltre SARSA(1) simula l'algoritmo Monte Carlo generalizzandolo per un backup incrementale, in modo che il metodo di controllo possa imparare immediatamente la situazione in corso, cambiare il proprio comportamento all'interno di uno stesso episodio in modo di essere così più reattivo. Nel capitolo sulla implementazione descriviamo la struttura delle funzioni approssimanti, delle basis functions scelte e i learning rate che risultano punti critici per il buon funzionamento dell'algoritmo.

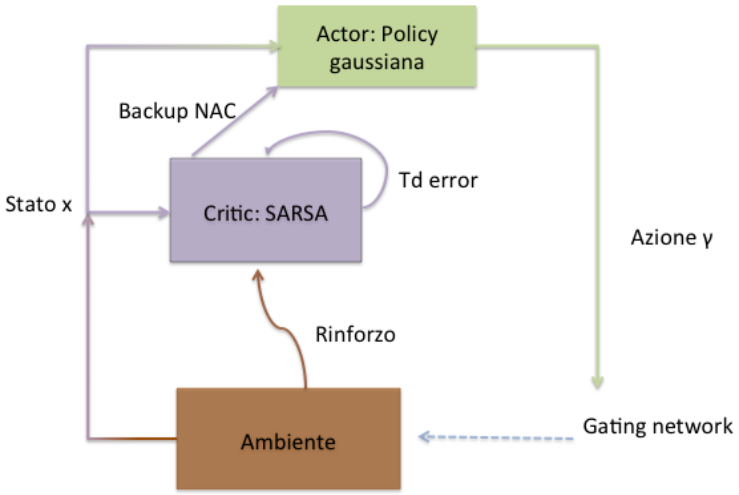

Figura 4.4: Struttura del modello della corteccia premotoria

4.4 Codifica degli ingressi e motivazioni

Il sistema di generazione del movimento necessita di un input corrispondente allo stato dell'ambiente esterno e del sistema interno per poter calcolare il movimento. I classici approcci di RL si basano sull'assunzione che il problema che affrontiamo sia un Markov Decision Process [23] che consiste di un insieme di stati, un insieme di azioni, una funzione di reward e una funzione di probabilità di transizione tra stati che cattura la dinamica del sistema. In questo caso il problema soddisfa la proprietà di Markov che afferma che la risposta dell'ambiente al tempo t+1 dipende solo dalla rappresentazione dello stato e della azione al tempo t, e non da informazioni aggiuntive su stati e azioni passate. In questo modo uno stato di Markov fornisce una statistica sufficiente per scegliere le azioni.
In generale in robotica, si riesce a trovare solo una approssimazione di uno stato di Markov. La nostra scelta dell'input quindi deve essere abbastanza ricca da contenere tutte le informazioni rilevanti senza degradare le performance dell'algoritmo di apprendimento. Infatti con l'aumentare del numero di dimensioni dello spazio degli stati, aumentano in modo esponenziale anche i dati e i calcoli per coprire completamente lo spazio degli stati e delle azioni. Nel nostro modello vogliamo una rappresentazione degli stati che ci permetta di avere numerose informazioni da modalità sensoriali differenti e mantenere ridotta la dimensionalità degli input. Per risolvere questo problema abbiamo collegato il nostro sistema all'architettura IDRA [44]. Essa è una modellizzazione del circuito corteccia-talamo-amigdala che crea una rappresentazione sensoriale bioispirata che soddisfa i requisiti definiti sopra e sviluppa autonomamente nuovi obiettivi in base all'esperienza del robot e dei suoi istinti innati. Per la fase di codifica sensoriale si ispira alla caratterizzazione della corteccia in base al metodo di apprendimento non supervisionato. Implementa in un modulo, chiamato modulo di categorizzazione, una funzione di riduzione di dimensionalità dell'input attraverso l'algoritmo Independent Component Analysis (Appendice A), replicando il meccanismo di statistical coding. Il segnale in entrata viene quindi proiettato nello spazio delle componenti indipendenti, riducendosi ad un insieme di coordinate su cui poi vengono costruite delle categorie per somiglianza applicando l'algoritmo di clustering k-Means. La rappresentazione dello stato in entrata al modello motorio è quindi un vettore delle distanze dai centri delle categorie formate con l'esperienza. Il sistema motorio si attiva solamente se lo stato attuale è interessante per robot, ovvero se è coerente con gli obiettivi sviluppati con l'esperienza dal robot. Questa ultima funzione è implementata nei rimanenti due moduli del talamo e amigdala, chiamati

rispettivamente modulo ontogenetico e filogenetico.Il primo valuta l'interesse generato autonomamente con l'esperienza, l'altro rispetto a degli istinti innati. Il componente che racchiude i moduli di categorizzazione e ontogenetico insieme si chiama modulo intenzionale.

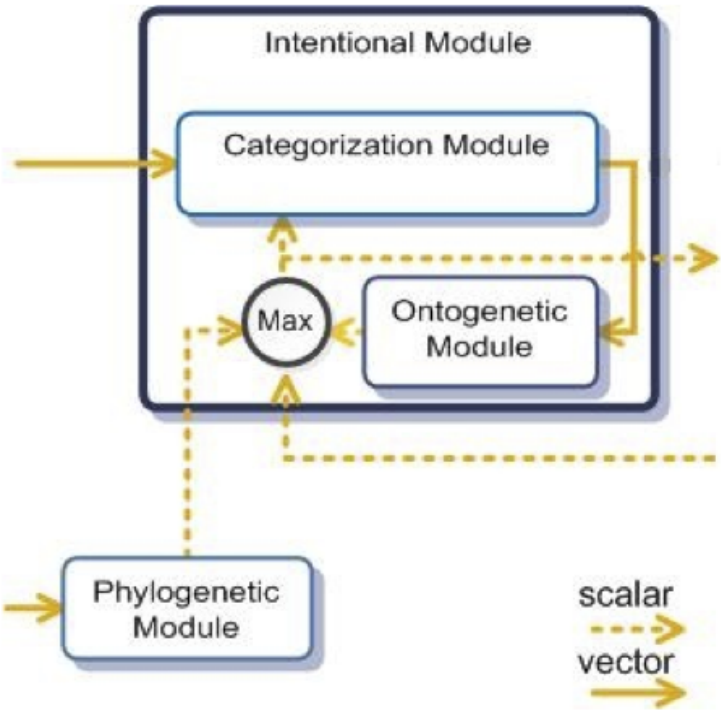

Figura 4.5: Modulo intenzionale

Esso può essere usato indistintamente per qualsiasi segnale, come la visione o l'olfatto. In entrata al nostro modello abbiamo inserito un modulo filogenetico e un modulo intenzionale che replica, nella fase di codifica degli ingressi, il funzionamento della corteccia parietale posteriore di integrazione di stimoli di diversa natura. Fornisce una rappresentazione unica dello stato a partire da un segnale visivo, filtrato come avviene nell'uomo, concatenato alle informazioni propriocettive. L'uscita quindi è un vettore di distanze dai centri delle categorie che abbiamo precedentemente appreso e un segnale di interesse per lo stato corrente.

Capitolo 5

Implementazione

In questo capitolo presentiamo le scelte implementative che hanno portato alla realizzazione del modello proposto. Vengono illustrate le principali formule utilizzate per la modellizzazione delle varie parti del sistema e le decisioni prese riguardanti i parametri aperti. Il modello è stato scritto nel linguaggio di programmazione Matlab, usando dove possibile le funzioni fornite dal linguaggio stesso, per poterlo testare sul robot Nao.

5.1 Dynamic Movement Primitives

Le DMP rappresentano l'elemento base del nostro movimento. Il modello proposto il questo lavoro di tesi impara a generare movimenti complessi a partire da un dataset di DMP che possono essere innate o acquisite. Esse non rappresentano traiettorie cinematiche fisse ma traiettorie che mantengono il comportamento qualitativo qualunque sia lo stato iniziale e finale del movimento. Esistono due tipi di DMP, discrete e ritmiche, di cui abbiamo gia dato una definizione. Descriviamo qui la loro formulazione matematica facendo riferimento al tipo discreto, implementato in questo lavoro e tratto in [5]. Esse sono modellizzate da due sistemi di equazioni differenziali, un sistema di trasformazione e un sistema canonico. Il primo è rappresentato dalle seguenti equazioni

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z),\ \tau \dot{y} = z + f \tag{5.1}$$

dove g è la posizione di arrivo del nostro movimento, α_z e β_z sono costanti, τ è la durata complessiva del movimento e y e la sua derivata corrispondono alla posizione e velocità generate dalle equazioni. Per una scelta opportuna

delle costanti e per $f = 0$, queste equazioni formano un sistema dinamico lineare globalmente stabile con g come unico punto attrattore. Affinchè questo sistema generi movimenti complessi viene introdotta una funzione non lineare f che dipende anche dal secondo sistema

$$\tau\dot{v} = \alpha_v(\beta_v(g - y) - v),\ \tau\dot{x} = v \tag{5.2}$$

Esso è un sistema del secondo ordine simile al precedente, ma è lineare e quindi la sua convergenza a g può essere garantita con una scelta di α_v e β_v tale per cui il sistema è smorzato. Questo sistema genera due quantità, una variabile di fase x e una di velocità v, che sono usate dalla funzione f

$$f(x, v, g) = \frac{\sum_{i=1}^{N} \psi_i w_i v_i}{\sum_{i=1}^{N} \psi_i},\ dove\ \psi_i = exp(-h_i(\frac{x}{g} - c_i)^2) \tag{5.3}$$

Questa funzione approssimante è rappresentata da basis functions normalizzate con parametrizzazione lineare. Il quoziente $\frac{x}{g} \in [0, 1]$ serve come variabile di fase per ancorare le basis function gaussiane, ognuna con un loro centro e una loro varianza, e la variabile v ha lo scopo di diminuire l'influenza della funzione f alla fine del movimento. I due sistemi insieme convergono asintoticamente all'unico punto attratore g, nonostante la presenza del termine f.

Le DMP vengono usate nel nostro modello per generare delle traiettorie angolari, in termini di posizione e non di velocità e accelerazioni, per i giunti del sistema robotico a partire da una posizione angolare iniziale. Per ogni grado di libertà abbiamo associato un sistema di trasformazione mentre il sistema canonico è unico per tutti i giunti in modo che si possa sincronizzare l'evoluzione temporale delle traiettorie.
Il sistema presenta dei metaparametri come il goal g, o la durata del movimento che possono essere modificati ma nella nostra implementazione abbiamo preferito fissarli a priori per semplificare il problema e concentrarci sulla composizione del movimento.
Affinche una DMP generi una traiettoria da noi specificata, i parametri della funzione f devono essere determinati. Essendo lineare nei parametri, molti algoritmi possono essere applicati per stimarli. La classe di algoritmi che abbiamo scelto di implementare corrisponde ad una di forma di apprendimento naturale dei movimenti che si verifica nell'uomo, che è l'apprendimento per imitazione. In [45] un robot umanoide è stato addestrato a colpire un pallina

con i movimenti tipici del tennis da esempi di movimenti umani catturati con un esoscheletro. Ciò è stato possibile applicando un algoritmo di regressione non parametrica [46]. Noi abbiamo adottato invece un algoritmo più semplice, fornendo al sistema di apprendimento una traiettoria desiderata $\dot{y}_{desiderata}$, fissata una durata del movimento, un goal e il numero di basis functions, posizionate in modo uniforme su tutto lo spazio della variabile di fase. A partire dalle equazioni descritte sopra otteniamo le seguenti formule

$$f = \tau \dot{y}_{desiderata} - z_{desiderata} \tag{5.4}$$

$$\tau \dot{z}_{desiderata} = \alpha_z(\beta_z(g - y_{desiderata}) - z_{desiderata}) \tag{5.5}$$

Per ottenere un valore della funzione f e applicare l'algoritmo abbiamo integrato con il sistema canonico specificando un intervallo di tempo, lo stesso con il quale abbiamo campionato la traiettoria da apprendere. A partire così dagli esempi (v, f) troviamo l'unica variabile di cui non conosciamo il valore, i parametri che caratterizzano il movimento.

5.2 La combinazione di primitive

Imparare a compiere un movimento corrisponde a trovare una policy che lega un vettore x rappresentante lo stato del sistema e dell'ambiente ad un vettore che rappresenta i comandi per far eseguire il movimento. Dato che noi ragioniamo in termini di traiettorie cinematiche, il vettore q in uscita dalla policy sarà una posizione, una velocità o una accelerazione per i giunti del robot.

$$q = \pi(x, t) \tag{5.6}$$

Apprendere questa policy è di solito complesso, quindi affrontiamo il problema riscrivendo l'equazione come composizione di basis function.

$$q = \pi(x, t) = \sum_{k=1}^{N} \pi_k(x, t) \tag{5.7}$$

Riduciamo il problema imparando una policy complicata come combinazione di policy più semplici, cioè primitive motorie. Dato un set di primitive

funzionali $\pi_{k,i}(x)$, tutte con lo stesso scopo i, definiamo una policy che combini il movimento di queste primitive in un nuovo movimento:

$$\pi(x) = \sum_{k=1}^{N_i} \frac{\gamma_k \pi_{k,i}(x)}{\sum_{h=1}^{N_i} \gamma_h} \tag{5.8}$$

Il risultato di questa operazione è una generalizzazione dei movimenti che hanno uno stesso scopo; per esempio posso definire un nuovo tipo di dritto nel gioco del tennis a partire da diversi esempi di tiri. Quando voglio costruire un movimento complesso posso voler combinare movimenti con scopi diversi. Per risolvere questo problema abbiamo pensato di sovrapporre i movimenti generalizzati da gruppi funzionali. La policy risultante, che modellizza il funzionamento della corteccia premotoria, è quindi

$$\pi(x) = \sum_{k=1}^{N_1} \frac{\gamma_k \pi_{k,1}(x)}{\sum_{h=1}^{N_1} \gamma_h} + ... + \sum_{k=1}^{N_j} \frac{\gamma_k \pi_{k,j}(x)}{\sum_{h=1}^{N_j} \gamma_h} \tag{5.9}$$

Il modello proposto quindi deve essere in grado generalizzare movimenti con lo stesso scopo, selezionare tra questi solo i movimenti utili e sovrapporre movimenti funzionalmente differenti per generare un nuovo movimento complesso. In entrata al modello abbiamo le traiettorie generate dalle DMP e in uscita abbiamo una traiettoria finale del movimento. Per apprendere questo modello bisogna trovare i valori ottimali dei parametri γ_i, ovvero i pesi della gating network della mixture of expert rappresentata dal modello.
Possiamo vedere il peso della primitiva i-esima come la probabilità di attivare la primitiva dato lo stato corrente

$$\gamma_i(x) = p(\pi_i|x) \tag{5.10}$$

Questa formulazione ci suggerisce di vedere i pesi come delle policy stocastiche stazionarie che rappresentano le policy da ottimizzare con l'algoritmo di RL descritto nel prossimo paragrafo.

5.3 L'apprendimento dei movimenti complessi

Nel nostro modello apprendere un movimento complesso significa imparare i pesi con cui valutare gli esperti della mixture of expert. Abbiamo pensato di modellizzare il peso i-esimo come una funzione approssimante lineare nei

parametri

$$\gamma_i(x) = \theta_i^T \phi_i(x) \tag{5.11}$$

dove θ_i è il vettore dei parametri e ϕ_i è il vettore delle basis functions. La scelta delle basis function è fondamentale per rappresentare bene la policy. Abbiamo scelto delle basis function gaussiane, in numero uguale alle primitive disponibili al sistema, con i centri corrispondenti agli stati tipici dove si attiva la primitiva i-esima, che si è rivelata una buona scelta anche in altri lavori sull'argormento [47]. La policy così risulta deterministica; a questa noi abbiamo scelto di aggiungere l'esplorazione per l'esecuzione dell'algoritmo di RL sommando un termine $\epsilon \sim N(0, \Sigma)$ in modo che la policy diventi di tipo stocastico

$$a_i = \theta_i^T \phi_i(x) + \epsilon \sim \pi_{i,\theta}(a|x) = N(a|\theta_i^T \phi_i(x), \Sigma), \; con\; a \in [0, 1] \tag{5.12}$$

L'algoritmo Actor-Critic che abbiamo deciso di implementare avrà quindi questa forma per l'Attore

$$\pi_{i,\theta}(a|x) = \frac{1}{\sqrt{2\pi\Sigma}} \exp(-\frac{(a - \theta_i^T \phi_i(x))^2}{\Sigma}) \tag{5.13}$$

Abbiamo decisono di usare SARSA(1) per il Critico e modellizzare la funzione Q che valuta le coppie $(stato, azione)$ utilizzando la "compatible function approximation" [30]

$$Q_w(x, a) = w^T \psi_\theta(x, a) \tag{5.14}$$

dove w è il vettore dei parametri, $\psi_\theta(x, a)$ è il vettore delle basis functions della funzione Q che sono chiamate score functions e hanno la seguente forma

$$\psi_{i,\theta}(x, a) = \frac{\partial}{\partial\theta} log \pi_{i,\theta}(a|x) \tag{5.15}$$

Utilizzando questa particolare forma per le basis function si può derivare una espressione di aggiornamento per l'Attore che porta alla definizione dell'algoritmo Natural Actor Critic descritto nel precedente capitolo

$$\theta_{t+1} = \theta_t + \beta_t w_t \tag{5.16}$$

L'algoritmo esegue i seguenti passi e viene replicato ed eseguito in parallelo per apprendere ogni peso.

$$
\begin{gathered}
w, \theta, z = 0 \\
scelgo\, azione\, a\, iniziale\, random\, nello\, stato\, x \\
ripeti \\
eseguo\, azione\, e\, ricevo\, reward\, r\, e\, stato\, y \\
scelgo\, azione\, a'\, dalla\, policy \\
eseguo\, SARSA(x, a, r, y, a', w, z)\, e\, aggioro\, w, z \\
calcolo\, \psi(x, a) \\
aggiorno\, i\, pesi\, \theta_{t+1} = \theta_t + \beta_t w_t \\
x = y \\
a = a' \\
fino\, a\, convergenza
\end{gathered}
\tag{5.17}
$$

Ci sono diversi parametri aperti in questo modulo. Le varianze delle basis functions della media parametrica dell'Attore, la varianza della policy dell'Attore e i due learning rate dell'Attore e del Critico. Per il buon funzionamento dell'algoritmo una prima scelta obbligata è quella di prendere il learning rate del Critico molto più alto di quello dell'Attore. Abbiamo scelto di prendere il learning rate del Critico costante, perchè sebbene il problema che il robot affronta possa essere stazionario, il problema che il Critico affronta, valutare una policy che continua a cambiare, non è stazionario. Anche il learning rate dell'Attore lo abbiamo fissato ad un valore costante perchè l'apprendimento avviene in contemporanea per tutti i pesi, il cui valore determina la bontà del movimento, quindi vi è la necessità di continuare ad apprendere in un ambiente in continua evoluzione. Per tutti i parametri aperti sono state fatte delle prove durante la fase di test per scegliere il valore più adatto al task considerato.
Come ultimo commento sull'algoritmo, i rewards ricevuti durante l'esecuzione del task, dato che l'algoritmo impara i pesi della gating network, sono pesati per il contributo che la primitiva i-esima ha avuto nella costruzione dell'azione

$$
R_i = R \frac{\gamma_i}{\sum_{j=1}^{N} \gamma_j} \tag{5.18}
$$

dove R è il reward complessivo derivante dall'esecuzione dell'azione e R_i è il reward della primitiva i-esima.

5.4 La codfica degli ingressi con IDRA

L'architettura IDRA fornisce una rappresentazione compatta dello stato in cui si trova il robot e un segnale che indica quanto questo stato sia per lui interessante. Per il nostro esperimento utilizziamo solo un modulo Intenzionale come ingresso ai moduli della corteccia premotoria e dei gangli, così da integrare le funzionalità di questa architettura e al tempo stesso non introdurre troppe complessità. Gli stimoli che riceve provengono da due modalità sensoriali, quella visiva e quella somatica. Il segnale visivo viene filtrato con un filtro logPolare, come avviene nell'uomo [8], e concatenato alle posizioni dei giunti a formare il segnale di ingresso del modulo Intenzionale. La funzione di riduzione di dimensionalità dell'input viene implementata nel modulo di categorizzazione, interno a quello Intenzionale, che esegue gli algoritmi ICA e K-means . L'Analisi delle Componenti Indipendenti permette al modulo di generalizzare la rappresentazione dell'ingresso, indipendentemente dal tipo di stimolo percepito. Nella fase di addestramento a priori vengono estratte le componenti indipendenti da una serie di ingressi dello stesso tipo che il modulo riceverà nella fase dopo l'addestramento, nel nostro caso corrispondono ad una serie di immagini e posizioni dei giunti concatenati. Nella fase successiva, ogni stato in cui si trova il robot (sempre descritto come concatenazione del vettore immagine con il vettore posizioni dei giunti) viene proiettato in questo spazio di componenti indipendenti secondo la seguente formula

$$\overline{W} = IC \,\mathrm{x}\, \bar{\mathrm{I}} \tag{5.19}$$

dove $\overline{W}$ è il vettore dei pesi risultante (il punto proiettato), IC è la matrice delle componenti indipendenti e $\overline{I}$ è il vettore in ingresso. Al punto proiettato è associato un interesse che può essere di natura innata (se proviene dal modulo filogenetico) oppure sviluppato con l'esperienza (se indicata invece dal modulo ontogenetico). Se questo interesse è sopra una certa soglia fissata a priori e se tutte le distanze dalle categorie già esistenti sono inferiori ad un'altra soglia, sempre fissata a priori, il punto viene aggiunto al gruppo più vicino tramite l'algoritmo K-means. Se, invece, almeno una delle distanze dai centroidi risulta essere sopra la soglia, il punto viene aggiunto all'insieme dei punti interessanti e viene ricalcolata l'assegnazione

alle categorie aggiungendo un nuovo gruppo, sempre tramite l'algoritmo di categorizzazione. Nel momento in cui l'interesse risulta essere sotto soglia, il sistema reputa il segnale in ingresso non interessante e, quindi, non viene eseguita la procedura di categorizzazione. L'uscita del modulo intenzionale infine è un vettore delle distanze dai cluster presenti. I moduli che corrispondono all'interesse innato e generato autonomamente sono il modulo filogenetico e ontogenetico rispettivamente, anche essi inclusi nel modulo Intenzionale. Il primo lo abbiamo implementato adattandolo al tipo di task su cui abbiamo testato il robot e lo illustriamo nel capitolo successivo, mentre il secondo non lo abbiamo implementato perchè avrebbe richiesto un ulteriore addestramento che esula dallo scopo di questa tesi. Il significato di avere un interesse nel nostro caso è quello di saper interpretare l'input e decidere se eseguire un movimento o meno ricreando un comportamento intelligente. Questo obiettivo può essere raggiunto anche con il solo modulo degli istinti innati.

Capitolo 6

Realizzazioni sperimentali e valutazione

Per testare il funzionamento del modello proposto abbiamo svolto due esperimenti sul robot NAO. Tutti e due gli esperimenti si basano sulla stessa tipologia di movimenti che il robot deve imparare a costruire a partire da un insieme di primitive motorie a disposizione. Il primo esperimento vuole valutare la capacità di apprendimento solo dei moduli di apprendimento motorio dando in ingresso uno stato semplificato. Il secondo vuole invece testare le prestazioni dell'intero modello proposto che ha IDRA come modulo iniziale.

6.1 Ambiente di lavoro e preparazione agli esperimenti

Tutti i moduli del modello sono stati scritti nel linguaggio di programmazione MatLab. L'implementazione del modulo delle DMP usata è quella fornita in [48]. Viene fornito anche il codice dell'apprendimento supervisionato per addestrare il modello al quale sono state apportate delle modifiche per adattare l'algoritmo ai nostri test. Il modulo per la composizione del movimento a partire dalle primitive, così come il relativo modulo di apprendimento sono stati scritti partendo da zero. Il codice di IDRA lo abbiamo invece ricavato in MatLab traducendolo dal linguaggio C#, come proposto in [1]. Per gli esperimenti abbiamo usato il robot NAO. E' un robot umanoide molto complesso sviluppato dalla Aldebaran Robotics. Ha 21 gradi di libertà e diversi tipi di sensori, tra i quali due camere per la visione, sensori tattili e microfono. Inoltre possiede un accelerometro, un girometro e quattro sensori di

prossimità a ultrasuoni che, insieme ai sensori di pressione sotto i piedi gli forniscono stabilità e capacità di posizionamento nello spazio.
.

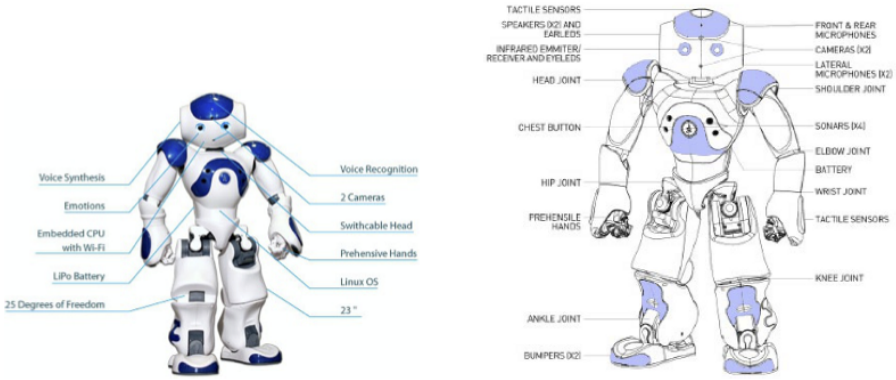

Figura 6.1: Il robot NAO

Abbiamo preferito testare il lavoro svolto su un robot reale, invece che al simulatore, per valutare le capacità del modello in situazioni reali senza alcune semplificazioni che possono essere presenti nella simulazione. Per di più, il simulatore software a disposizione era molto limitato nella scelta degli oggetti con cui il robot avrebbe dovuto interagire. Insieme al robot sono fornite delle API per chiamare dal codice sviluppato le funzioni necessarie per interagire con il NAO, come ad esempio quelle che restuiscono ciò che il NAO vede o quelle per farlo muovere. Dato che l'ultima versione del firmware del robot non supporta ancora le API per MatLab, abbiamo deciso di utilizzarne una versione antecedente, preferendo la semplicità di implementazione di MatLab.
Come compito per il primo esperimento abbiamo pensato di chiedere al robot di coprire con un bicchiere posto nella sua mano sinistra una pallina posta di fronte a lui; Abbiamo posizionato il robot seduto ad un tavolo in modo che riuscisse, usando solo il braccio sinistro per semplicità, a raggiungere la pallina. Il modello controlla il braccio sinistro del robot, composto da quattro gradi di libertà, due per la spalla e due per il gomito. Lo scopo è di testare la sua capacità di generare un movimento complesso a partire da movimenti semplici quali possono essere i riflessi presenti nell'uomo.
Il dataset di primitive creato è composto da sette riflessi neonatali e due primitive acquisite di rotazione della mano: il riflesso tonico asimmetrico cervicale, due versioni leggermente diverse del riflesso di Moro, il riflesso

spinale di Galant, due varianti del riflesso di nuoto e il riflesso di caduta in avanti. Alcuni sono stati descritti nel capitolo due, altri possono essere trovati in [49]. Si ipotizza che il robot abbia acquisito due primitive di rotazione della mano, una che ruota solo la mano, l'altra che piega leggermente anche l'avambraccio. Il polso non è attuato quindi la rotazione avviene al livello del gomito. La presenza di diverse versioni di alcuni riflessi riflette il comportamento umano; si è infatti osservato che, a seguito dello stesso stimolo, vengono eseguiti movimenti leggermente differenti ma qualitativamente simili in relazione alla posizione del corpo.

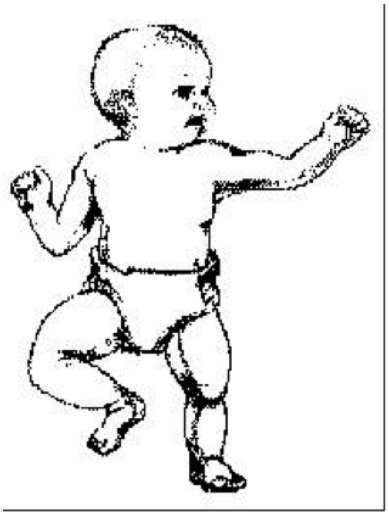

Figura 6.2: Il riflesso tonico asimmetrico cervical in un bambino

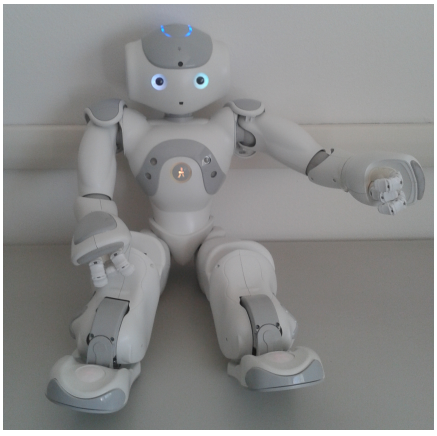

Figura 6.3: Il riflesso tonico asimettrico cervicale replicato dal NAO

La fase di preparazione agli esperimenti consiste nella creazione del dataset di primite motorie. Come prima cosa abbiamo registrato le traiettorie desiderate, corrispondenti ai movimenti descritti sopra, muovendo fisicamente il braccio del robot con le nostre mani e registrando le posizioni dei quattro giunti del suo braccio sinistro a intervalli di tempo regolari di 0.4 secondi. Per fare ciò abbiamo utilizzato il software Choreographe, fornito con il NAO. In seguito abbiamo eseguito l'apprendimento del modello delle DMP a partire dalle traiettorie appena registrate. Abbiamo fissato la durata del movimento a 3 secondi, mentre abbiamo scelto il numero di basis function per il termine nonlineare f empiricamente, impostando il valore che produceva una traiettoria più simile a quella desiderata. Una volta registra-

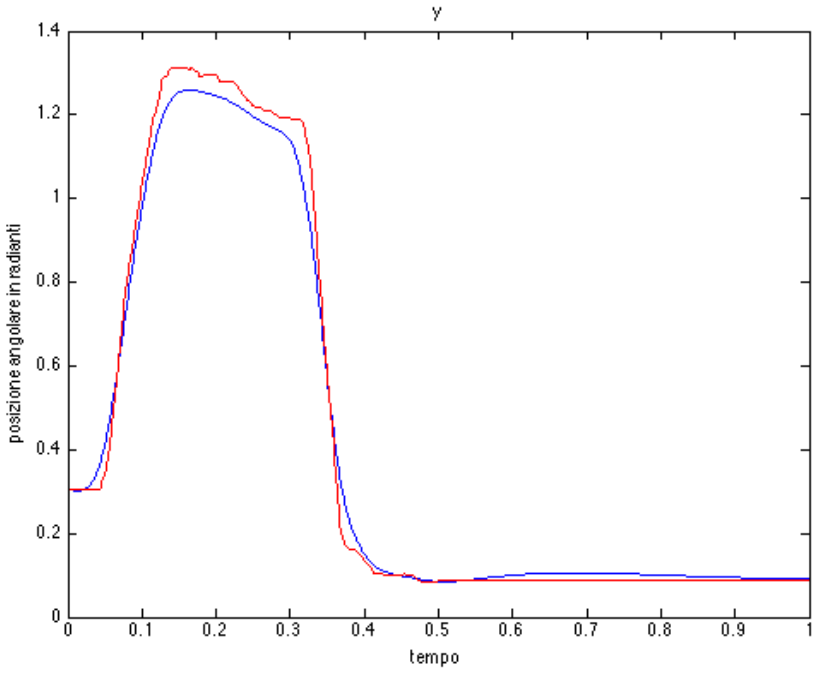

Figura 6.4: Traiettoria di una DMP che approssima il movimento di rotazione della spalla nella prima versione del riflesso di nuoto

to il dataset e posizionato il robot come descritto sopra, abbiamo eseguito il codice nelle due versioni descritte in precedenza; il computer che abbiamo impiegato per gli esperimenti ha questa configurazione:

- processore Intel Core i7-920
- scheda grafica Nvidia GTS250 1GB

- ram 4GB
- hard disk 1TB 7200rpm

6.2 Primo esperimento: la corteccia premotoria e i gangli

Il primo esperimento ha lo scopo di testare la capacità di generazione e di apprendimento di nuovi movimenti senza introdurre complessità riguardanti la codifica bioispirata dello stato dell'ambiente e del corpo. Il modello che controlla il robot è quindi formato dai soli moduli della corteccia premotoria e dei gangli che ricevono in ingresso uno stato composto dalle coordinate cartesiane (x, y, z) della pallina rispetto al torso del robot e le quattro posizioni angolari iniziali del braccio sinistro. Un test è composto da una serie di prove in cui il robot cerca di coprire la pallina che rimane fissa nella stessa posizione. Lo stato iniziale in cui si trova il robot è ogni volta diverso perchè la posizione del braccio viene impostata in modo casuale a partire da un set di posizioni iniziali possibili. Il test continua per qualche iterazione anche dopo che il robot riesce a prendere la pallina, in questo modo si riesce ad evidenziare meglio che l'algoritmo raggiunge la stabilità. Vengono eseguite più prove con lo stesso setting per poi cambiare la posizione della pallina e rieseguire dall'inizio il test. I learning rate scelti per questa tipologia di esperimenti sono inizializzati al valore di 0.85 costante per il Critico e 0.35 per l'Attore.

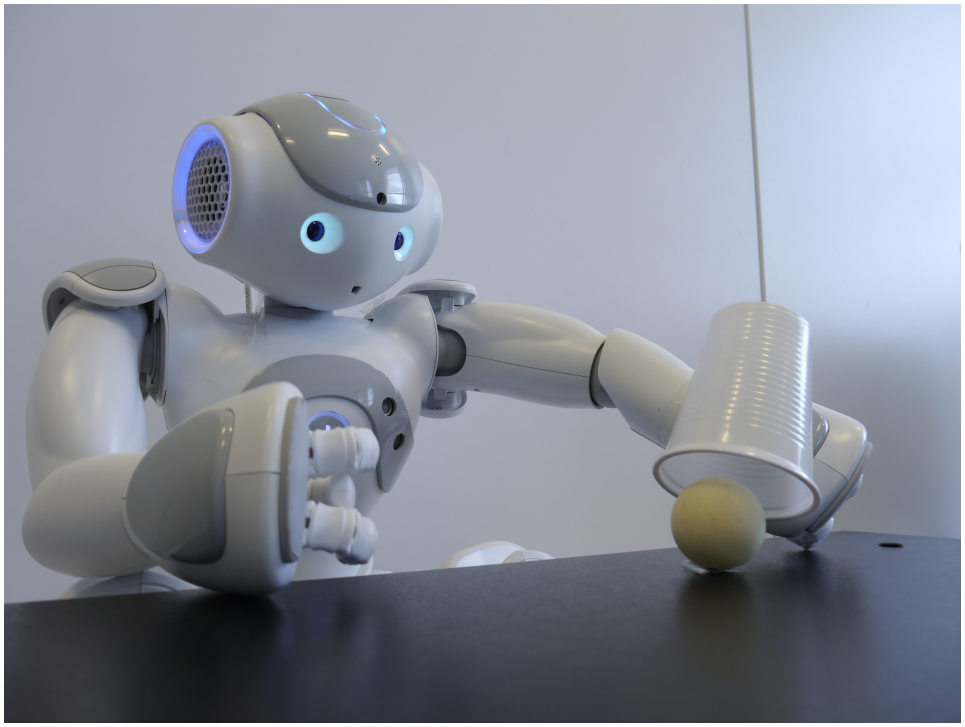

Figura 6.5: Il robot NAO

Per valutare le prestazioni del modello proposto abbiamo registrato durante i test l'evoluzione del reward. Il reward si compone di due quantità: una componente cartesiana che valuta la posizione dell'end effector nelle tre coordinate cartesiane e una angolare che tiene conto dell'orientamento della mano.

$$R = r_{cartesiano} + r_{angolare}$$

$$r_{cartesiano} = \exp(distanza^{1/3}_{cartesiana}) \tag{6.1}$$

$$r_{angolare} = |distanza_{angolare}| \tag{6.2}$$

Il reward cartesiano misura l'esponenziale della distanza tra l'end effector e la pallina, di cui prima viene fatta la radice cubica. Abbiamo eseguito questo passaggio per enfatizzare l'importanza di questa distanza nel processo di apprendimento. Il reward angolare misura invece il valore assoluto della distanza angolare tra la mano del robot e un'orientamento angolare, tale per cui la pallina viene coperta, che viene mostrato al robot all'inizio di ogni test. Le primitive motorie memorizzate che corrispondono ai riflessi neonatali eseguono tutte, in modi differenti, dei movimenti di estensione e le possiamo raggruppare in un unico gruppo di primitive funzionali con uno scopo di reaching senza rotazione nello spazio. Le altre due primitive invece formano un altro gruppo che si occupa solo della rotazione della mano. Abbiamo pensato quindi di differenziare i reward per questi due gruppi di primitive: il reward cartesiano contribuisce all'apprendimento del primo gruppo, mentre il reward angolare a quello del secondo. Ogni gruppo funzionale controlla l'andamento di una variabile. Ipotizziamo quindi che il contributo di un gruppo sia indipendente dall'altro. Per ogni peso i della gating network il reward è quindi:

$$r_{cartesiano,i} = \frac{r_{cartesiano}\gamma_i}{\sum_{j=1}^{N_1}\gamma_j} \tag{6.3}$$

oppure

$$r_{angolare,i} = \frac{r_{angolare}\gamma_i}{\sum_{j=1}^{N_2}\gamma_j} \tag{6.4}$$

Nella figura sottostante presentiamo l'andamento del reward R di un esperimento.

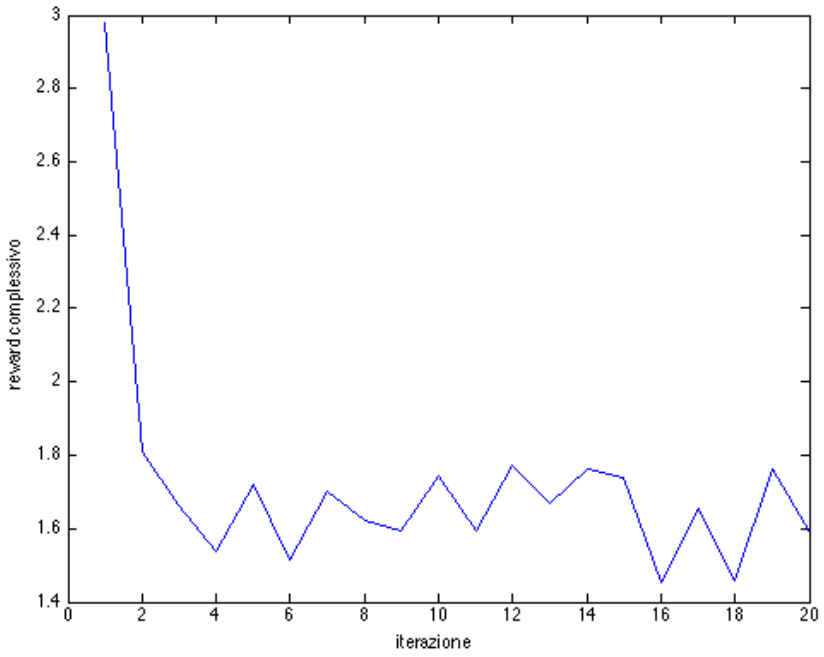

Figura 6.6: Andamento del reward complessivo per la prova numero 6

Possiamo osservare come il reward diminuisca con l'aumentare delle iterazioni dell'algoritmo, numero che corrisponde alle prove di movimento del robot. Tutti gli esperimenti presentano un andamento simile, mostrando che l'algoritmo sta effettivamente apprendendo il movimento corretto da eseguire, minimizzando, verso un valore fisso, la distanza tra la mano e la pallina. Nel grafico riportato è interessante notare l'andamento altalenante di questo reward. Questo comportamento può derivare dalla scelta della policy gaussiana che ha una media parametrica che viene adattata e una varianza Σ fissata a priori. In questo esperimento era pari a 0.4. Il suo ruolo è quello di permettere l'esplorazione dello spazio delle azioni. Maggiore è la varianza, maggiore sarà l'esplorazione. Anche se l'algoritmo aggiorna la media della policy verso un valore che minimizza la distanza, la varianza continuerà a far oscillare l'azione scelta intorno alla media con la stessa probabilità, al contrario di altri algoritmi che, utilizzando una politica di esplorazione ϵ-greedy, si stabilizzano su un valore (sub-)ottimo. Nei nostri esperimenti

abbiamo osservato che il valore migliore per la varianza era compreso tra 0.3 e 0.5. Scegliere una varianza troppo piccola significava non esplorare, sceglierla troppo alta rallentava l'apprendimento. Immaginiamo che le perfomance diminuiscano perchè l'alta esplorazione è presente per i pesi di tutte le nove primitive che entrano nella composizione del movimento finale, che non riescono così a generare un movimento buono con dei valori dei pesi che si discostano molto da quelli ottimali. Mostriamo anche l'andamento dei $reward_{cartesiano}$ e $reward_{angolare}$ per lo stesso esperimento:

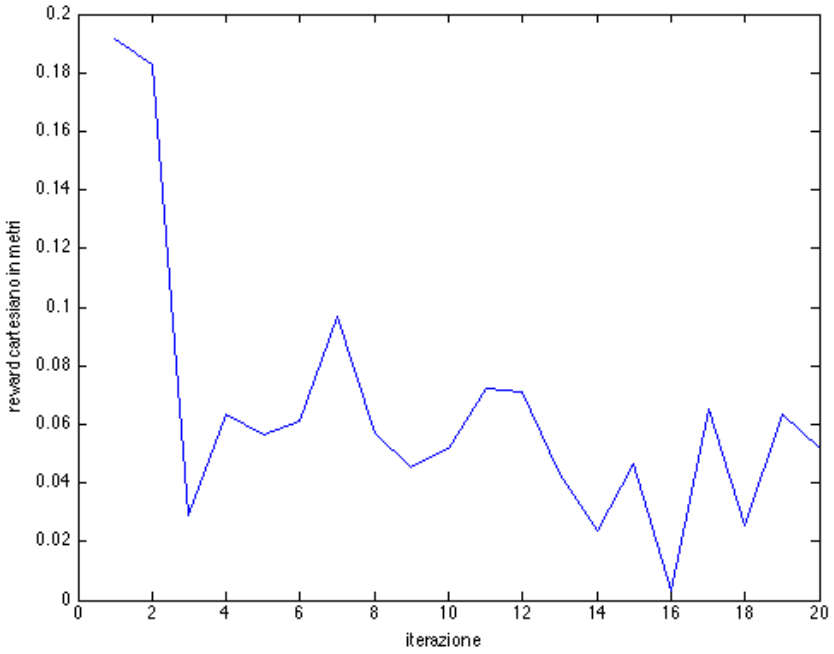

Figura 6.7: Andamento del reward cartesiano per la prova numero 6

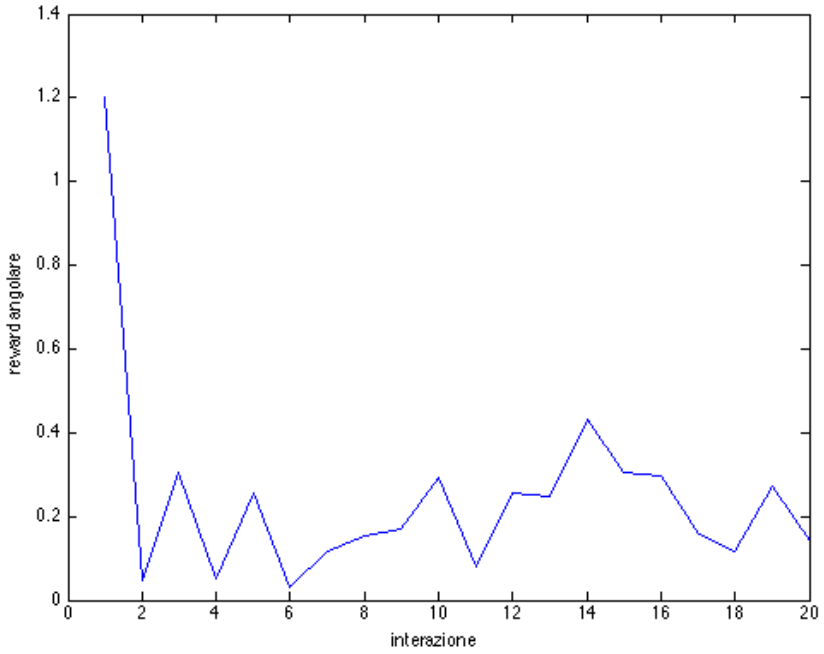

Figura 6.8: Andamento del reward angolare per la prova numero 6

Valutando questo primo esperimento da un punto di vista qualitativo e non più quantitativo, il robot è riuscito a coprire la pallina anche se la distanza, espressa in metri quella cartesiana e in radianti quella angolare, non tocca lo zero. Questo perchè il robot non deve toccare la pallina ma coprirla con un bicchiere che ha una apertura tale da permettergli qualche centrimetro di errore. Inoltre abbiamo osservato che le misurazioni della posizione del braccio, eseguite con NAO, non restituivano sempre valori precisi. Dal punto di vista biologico, si osserva nell'uomo un meccanismo di feedback per correggere in tempo reale le traiettorie [50]. Questa funzionalità non è considerata in questa tesi, però, nell'ottica di un futuro sviluppo di un modello bioispirato più completo, essa convalida la valutazione positiva delle prestazione del nostro sistema.
Il valore dei pesi medi per le 9 primitive, nell'ordine con cui sono state introdotte nel capitolo precedente, è:

0.45	0.413	0.082	0.452	0.85	0.12	0.847	0.63	0.2
Cervicale	Galant	Moro1	Moro2	Caduta	Nuoto1	Nuoto2	Rotaz1	Rotaz2

Si osserva che il movimento finale è composto da quasi tutte le primitive con diverso peso. Le terza primitiva è stata esclusa dalla generazione del movimento, così come la primitiva numero sei. Il sistema è stato così in grado di selezionare le primitive utili e usarle per combinare il movimento. Mostriamo l'andamento del reward complessivo mediato su 10 esperimenti in figura 6.9.

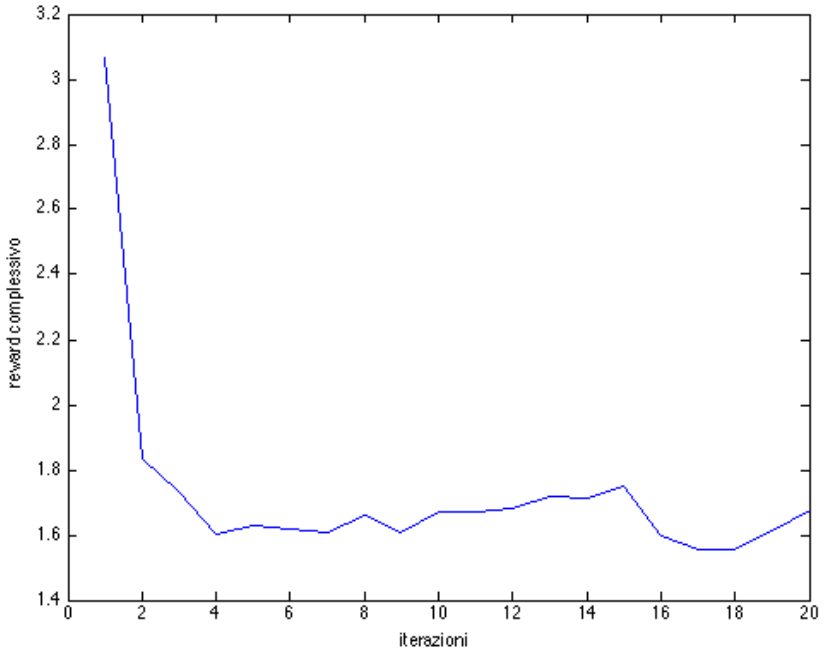

Figura 6.9: Andamento del reward complessivo medio

Precisiamo che il robot è riuscito a completare il test nella maggior parte delle situazioni proposte, ma non per tutte le posizioni della pallina. Immaginiamo che questo sia dovuto alla mancanza di un sistema di adattamento per le traiettorie delle singole primitive motorie che, nel nostro modello, possono generare movimenti complessi limitati ad una combinazione convessa dei movimenti base.

6.3 Secondo esperimento: il modello completo

Il secondo esperimento vuole valutare la capacità di apprendimento motorio sfruttando una codifica dell'input bioispirata. Abbiamo aggiunto ai due moduli del test precedente anche l'architettura IDRA personalizzandola per il nostro esperimento. In particolare abbiamo integrato nel nostro modello un modulo Intenzionale, che contiene un modulo di categorizzazione, e un modulo filogenetico per valutare l'interesse per lo stato. L'input del modello è costituito dalla posizione dei giunti del braccio sinistro del robot e dall'immagine presa dalla camera del NAO, dopo averla filtrata con un filtro

logPolare che abbiamo aggiunto all'entrata del modulo di categorizzazione.

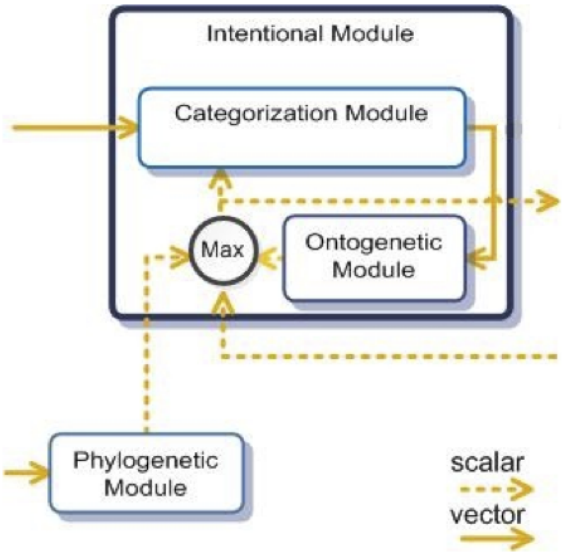

Figura 6.10: Modulo Intenzionale

Il modulo di categorizzazione deve essere addestrato a priori affinchè apprenda le componenti indipendenti dove proiettare l'input del sistema. Nella fase di addestramento abbiamo bisogno di due dataset per poter costruire l'input da dare ad ICA. Il primo contiene una serie di posizioni angolari del braccio del robot, che abbiamo deciso di rilevare muovendone in modo pseudo-random il braccio, mantenedone però la posizione della mano in un'area circoscritta definita a priori. Il secondo invece contiene un insieme numeroso di immagini naturali che provengono dall'ambiente di lavoro del robot. Ciò che è importante nella costruzione dei dataset è che gli esempi di addestramento siano dello stesso tipo dell'input che riceverà poi il robot, anche se non saranno effettivamente gli stessi. Il dataset delle immagini è stato reperito in [1]. Il modulo filogenetico è stato modellizzato in modo che restituisca un segnale di interesse tanto più alto quanto la pallina è vicina al robot. In questo modo il robot sarà interessato a giocare con una pallina che riesce a raggiungere. Per trovare la posizione della pallina abbiamo usato una funzione di MatLab che restuisce le coordinate in pixel della sfera a partire dalla visione del NAO.
Successivamente all'addestramento, il funzionamento dell'esperimento e il suo setup è il medesimo del precedente; anche per questo abbiamo regi-

strato l'andamento del reward. Il reward complessivo medio su 6 prove è riportato nella figura sottostante:

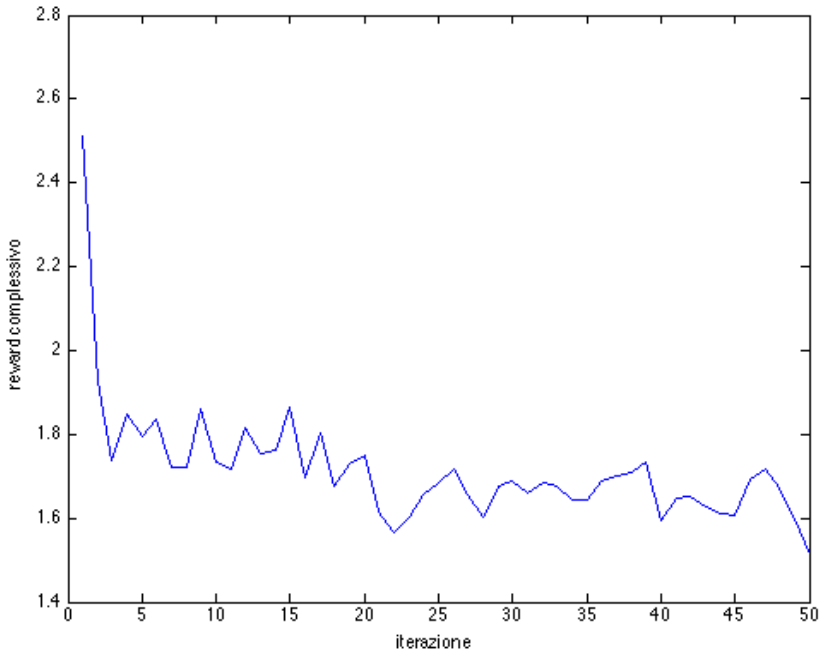

Figura 6.11: Andamento del reward complessivo medio

Osservando il grafico non si notano differenze sostanziali con il primo set di esperimenti. Anche in questo caso il robot è riuscito quasi tutte le volte a completare il task, iniziando il movimento solo per i trial dove il segnale di interesse era sopra la soglia di attivazione. Stando alle nostre ipotesi iniziali, ci saremmo aspettati un miglioramento delle prestazioni derivanti dal fatto che IDRA crea una rappresentazione più efficiente degli stati che poteva aiutare l'apprendimento delle funzioni approssimanti impiegate nell'algoritmo. Pensiamo che lo stato semplificato fornito per il primo set di esperimenti sia risultato equivalente al contributo di IDRA in questo secondo set. Certamente però in situazioni reali più complesse, per quali non è possibile creare in modo semplice uno stato ad hoc di ridotta dimensionalità, il contributo di IDRA diventa di fondamentale importanza; per di più se pensiamo al prossimo obiettivo di integrare più modalità sensoriali per la creazione di stati più complessi e più ricchi di informazioni.

Capitolo 7

Conclusioni e sviluppi futuri

Con questo lavoro di tesi vogliamo proporre un modello bioispirato per l'apprendimento e la pianificazione del movimento. In particolare il modello è stato sviluppato indirizzando le problematiche derivanti dai nuovi sistemi robotici, sempre più complessi da controllare, e dalle nuove situazioni in cui questi robot devono agire. Nell'ottica di futura integrazione nella società tra robot ed esseri umani, una caratteristica fondamentale per i robot è la capacità di muoversi eseguendo movimenti che non sono stati memorizzati a priori, per di più in un ambiente di cui non si conoscono tutte le dinamiche. Questa ricerca si è proposta di progettare ed implementare un primo modello motorio che prenda ispirazione dalla biologia per avvicinarsi alle capacità umane.
Dalla letteratura riportata nel capitolo sullo stato dell'arte abbiamo evidenziato alcuni processi biologici che intervengono nella costruzione del movimento. Il primo di questi è la reazione agli stimoli, che abbiamo visto essere composto da diverse fasi, con lo scopo di creare una rappresentazione compatta integrata di più modalità sensoriali. Abbiamo implementato questa soluzione integrando il nostro modello con l'architettura IDRA, di cui abbiamo personalizzato i moduli che riducono la dimensionalità dell'input e che generano un interesse verso una situazione. L'interesse è il collegamento tra l'architettura cognitiva e il sistema motorio. Risulta fondamentale abilitare i sistemi robot autonomi con questa caratteristica per avere una interazione naturale con l'uomo. Il modello motorio rispecchia i processi di generazione del movimento, secondo cui il movimento è rappresentato nel cervello a diversi livelli di astrazione, e gli elementi dei livelli più bassi vengono usati per costruire il movimento rappresentato dai livelli più in alto. In particolare noi ci concentriamo sull'astrazione delle primitive motorie e sui movimenti derivanti dalla combinazione di queste, che sono funzioni di

input senso-motori e non di segnali associativi e motivazionali più complessi. Abbiamo integrato nel nostro sistema il modello delle DMP per rappresentare le primitive motorie e abbiamo sviluppato un modello per combinarle con il relativo algoritmo di apprendimento. La soluzione proposta per questa fase consiste nella progettazione di una mixture of experts, che ha lo scopo di generalizzare movimenti funzionalmente simili, e di un algoritmo Actor-Critic per apprendere i pesi della relativa gating network.
I test sono stati condotti con l'ausilio del robot umanoide NAO, che doveva imparare a comporre un movimento a partire da un dataset di primitive motorie. I risultati dell'esecuzione del task assegnato sono stati simili nelle due versioni dell'esperimento e sono risultati soddisfacenti all'incirca nell'80% delle volte. Il robot impara a generare un movimento per diverse posizioni della pallina e mostra un comportamento più stabile in media dopo dieci tentativi. Essendo che il sistema di generazione del movimento crea combinazioni convesse di gruppi di primitive funzionali con lo stesso scopo e poi le sovrappone per formare il movimento finale, la scelta del dataset di primitive risulta un elemento critico del sistema. Non si possono creare tutti i movimenti ammissibili dalla meccanica del robot ma solo quelli che emergono dalle primitive motorie nel modo appena descritto. Inoltre la scelta del dataset influsce anche sulla velocità di apprendimento dell'algoritmo proposto, in quanto un numero alto di primitive può aiutare a generare un maggior numero di movimenti però nello stesso tempo rallenta l'apprendimento della rete che cerca di massimizzare le performance in una dinamica complessa e instabile soprattutto nella fase iniziale. Nei nostri esperimenti abbiamo scelto un dataset di nove primitive pensando che fosse un buon tradeoff. Pensiamo che l'algoritmo possa esprimere prestazioni migliori con una scelta più accurata del dataset, non solo nel numero ma anche nella diversità del movimento.
Non è comunque necessario che il sistema esibisca una elevata precisione in questa fase di composizione del movimento, in quanto uno sviluppo naturale del modello potrebbe essere l'aggiunta di un meccanismo di feedback per aggiustare in tempo reale il movimento, così come avviene nell'uomo [50]. Una idea sarebbe quella di aggiungere un termine di coupling nella generazione delle traiettorie delle DMP per ottenere un comportamento reattivo.

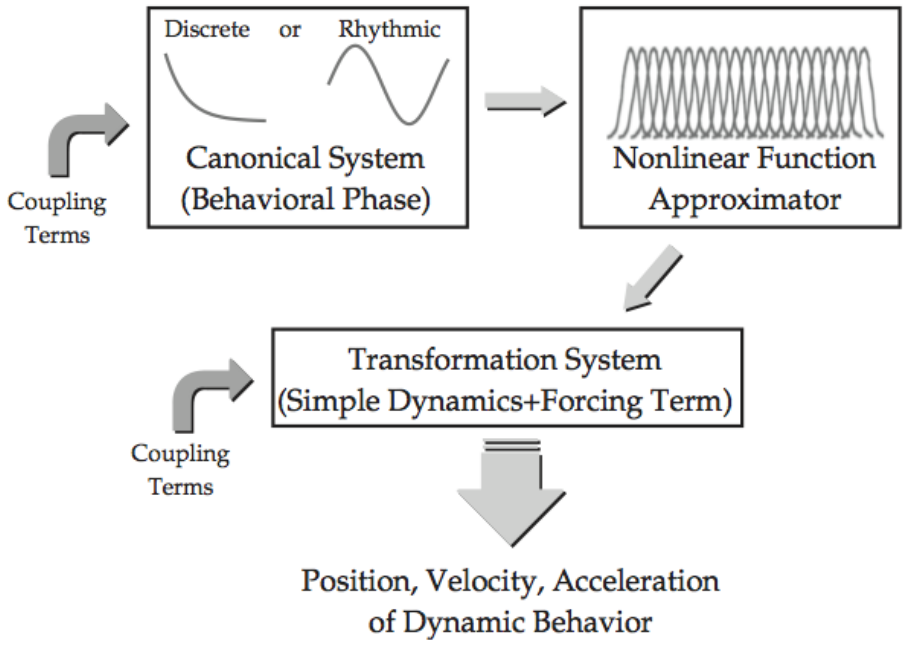

Figura 7.1: Modello delle DMP con termine di coupling

Per esempio si potrebbe volere aggiustare il movimento in relazione alla distanza stimata da un ostacolo che si vuole evitare o da un punto che si vuole raggiungere. La progettazione di questo meccanismo potrebbe includere una modellizzazione di altre aree del cervello come il cervelletto che svolge anche la funzione di trasformazione di coordinate [11].
Un altro modo per ottenere un adattamento del movimento fuori dall'esecuzione dell'algoritmo proposto potrebbe essere la progettazione di un sistema di adattamento dei metaparametri delle DMP. Abbiamo sviluppato un primo semplice algoritmo Actor-Critic per adattare il goal delle primitive, nella versione descritta in [23], basato sulla costruzione di una funzione di valutazione del solo stato, e non le coppie $(stato, azione)$ come avviene nell'algoritmo sopra. Abbiamo pensato ad una valutazione dello stato perchè risulta più stabile in apprendimenti dove la distribuzione dei reward presenta delle grandi aree con pessime prestazioni e solo limitate zone con reward buoni. Questo è il comportamento che ci aspettiamo si verifichi quando uniamo un processo di apprendimento che va a modificare i metaparametri di un altro processo. I risultati dell'applicazione di questo modello ai goal di una sola primitiva ha dato però comunque un comportamento distruttivo dell'apprendimento della gating network vanificando così l'apporto di entrambi. Crediamo che sia un punto su cui poter lavorare per ottenere dei

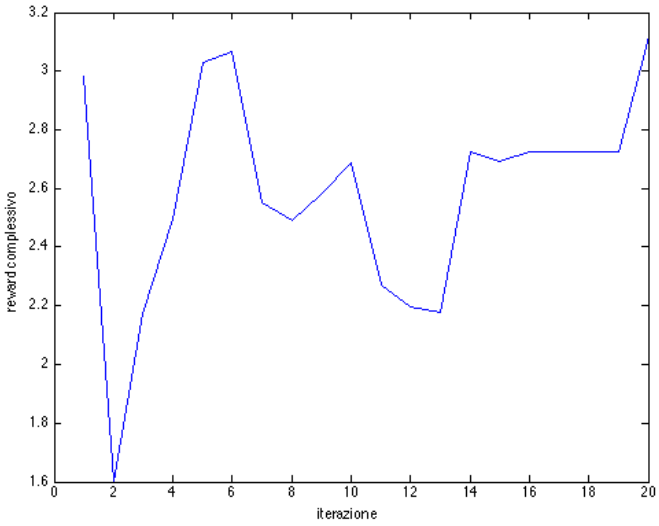

Figura 7.2: Comportamento distruttivo dell'apprendimento dei metaparametri

movimenti meno rigidi ma adattabili ancora di più ad altre situazioni.

Questi sono solo alcuni dei possibili sviluppi futuri che il modello proposto in questo lavoro permette di integrare. Crediamo che il lavoro svolto sia una buona base per la costruzione di una architettura motoria più completa. L'integrazione con IDRA ha mostrato il naturale collegamento tra i due sistemi e altri test in situazioni con stati più complessi possono mettere in mostra le reali capacità del sistema complessivo.

Bibliografia

[1] A. M. Franchi, "A dynamic network model simulating brain plasticity for a robotic bioinspired intentional architecture: implementation and test," Master's thesis, Politecnico di Milano, 2012.

[2] J. Konczak, "On the notion of motor primitives in humans and robots," *Proceeding of the fifth Workshop on Epigenetic Robotics*, 2005.

[3] R. W. Paine and J. Tani, "Adaptive motor primitive and sequence formation in a hierarchical recurrent neural network," *Neural Networks 17*, 2004.

[4] C. Alessandro, "Muscle synergies in neuroscience and robotics: from input-space to task-space perspectives," *Front. Comput. Neurosci.*, 2013.

[5] S. Schaal and J. Peters, "Control, planning, learning, and imitation with dynamic movement primitives," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2003.

[6] E. Bizzi and F. A. Mussa-Ivaldi, "Toward a neurobiology of coordinate transformations," *The Cognitive Neurosciences*, 2013.

[7] K. Doja, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" *Neural Networks 12*, 1999.

[8] F. Gallo and F. Ferrini, "Odin: Un sistema cognitivo bioispirato per l'apprendimento sensomotorio," Master's thesis, Politecnico di Milano, 2013.

[9] D. Angelaki, *The Cognitive Neurosciences.* The MIT press, 2013, ch. Multisensory Integration for Heading Perception in Macaque Visual Cortex, pp. 499–510.

[10] R.Kandel and J. Schwartz, *Principles of Neural Science Fourth Edition.* McGraw-Hill, 2000.

[11] K. Doja, "Complementary roles of basal ganglia and cerebellum in learning and motor control," *Current Opinion in Neur*, 2000.

[12] R. Pfeifer and M. Lungarella, "Self-organization, embodiment, and biologically inspired robotics," *Science*, 2007.

[13] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution," *Neural Comput.*, 1995.

[14] D. Lee and S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature 401*, 1999.

[15] E. Todorov, *Bayesian brain: probabilistic approaches to neural coding.* MIT press, 2006, ch. Optimal Control Theory, pp. 269–298.

[16] P. E. Latham, S. Deneve, and A. Pouget, "Encient computation and cue integration with noisy population codes," *Nature Neuroscience 18*, 2001.

[17] A. Schwartz and A. Georgopoulos, "Neuronal population coding of movement direction," *Science*, 1986.

[18] M. Carandini and D. J. Heeger, "Normalization as a canonical neural computation," *Nature Neuroscience*, 2012.

[19] W. Bialek and E. Schneidman, "Synergy, redundancy, and independence in population codes," *The Journal of Neuroscience*, 2003.

[20] D. Sparks and W. Kristan, *Neurons, Networks, and Motor Behavior.* MIT Press, 1997, ch. The role of population coding in the control of movement, pp. 21–32.

[21] P. Latham, B. Averbeck, and A. Pouget, "Neural correlations, population coding and computation," *Nature*, 2006.

[22] R. Bellman, *Dynamic programming.* Princeton University Press, 1957.

[23] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction.* The MIT press, 1998.

[24] S. Schaal, *Adaptive Motion of Animals and Machines.* Springer, 2006, ch. Dynamic Movement Primitives: A Framework for Motor Control in Humans and Humanoid Robotics, pp. 261–280.

[25] N. Schweighofer and M. A. Arbib, "Role of the cerebellum in reaching movements in humans." *Eur J Neurosci*, 1998.

[26] J. F. Kalaska, *Motor Control: Concepts and Issues.* John Wiley & sons, 1991, ch. What parameters of reaching are encoded by discharges of cortical cells?, pp. 307–330.

[27] N. Bernstein, *The control and regulation of movements.* Pergamon Press, 1967.

[28] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control — a unifying view," *Progress in Brain Research*, 2007.

[29] J. Peters and D. Nguyen-Tuong, "Model learning in robotics: A survey," *Cognitive Processing*, 2011.

[30] C. Szepesvari, *Algorithms for Reinforcement Learning.* Morgan & Claypool, 2010.

[31] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The I*, 2013.

[32] V. Konda, "Actor-critic algorithms," *Journal on Control and Optimization*, 2000.

[33] J. Peters and S. Schaal, "Natural actor-critic," *NIPS*, 2008.

[34] P. Thomas and W. Dabney, "Projected natural actor-critic," *NIPS*, 2013.

[35] E. R. Kandel, J. Schwartz, and T. M. Jessell, Eds., *Principles of Neural Science.* McGraw Hill, 2000.

[36] A. Hyvarinen and E. Oja, "Independent component analysis: Algorithms and applications." *Neural Networks*, 2000.

[37] K. Doja, "Multiple representations and algorithms for reinforcement learning in the cortico-basal ganglia circuit," *Current Opinion in Neu*, 2011.

[38] S. J. Thorpe, "Spike arrival times: A highly efficient coding scheme for neural networks." *Parallel processing in neural systems*, 1990.

[39] S. Amari, "Natural gradient works efficiently in learning," *Neural*, 1998.

[40] C. M. Bishop, *Neural networks for pattern recognition.* New York: Oxford University Press,, 1995.

[41] E. A. Theodorou, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, 2010.

[42] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *NIPS*, 2008.

[43] R. Williams, "Simple statistical gradient- following algorithms for connectionist reinforcement learning," *Machine Learning*, 1992.

[44] M. Burrafato and L. Florio, "A cognitive architecture based on an amygdala-thalamo-cortical model for developing new goals and behaviors: application in humanoid robotics," Master's thesis, Politecnico di Milano, 2012.

[45] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *International Conference on Robotics and Automation*, 2002.

[46] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional spaces," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[47] K. Mülling and J. Kober, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, 2013.

[48] [Online]. Available: http://www-clmc.usc.edu/ sschaal/

[49] L. Camaioni and P. D. Blasio, *Psicologia dello sviluppo.* Il Mulino, 2007.

[50] R. Ajemian, "A theory for how sensorimotor skills are learned and retained in noisy and nonstationary neural circuits," *PNAS*, 2013.

Appendice A

Estratti di codice

In questa sezione vengono riportati alcuni estratti del codice sorgente del programma scritto che implementa il modello corteccia-gangli. Vengono riportate solo le parti più interessanti che definiscono le fasi dell'algoritmo viste nei capitoli precedenti. Riportiamo anche uno schema a blocchi della struttura del modello:

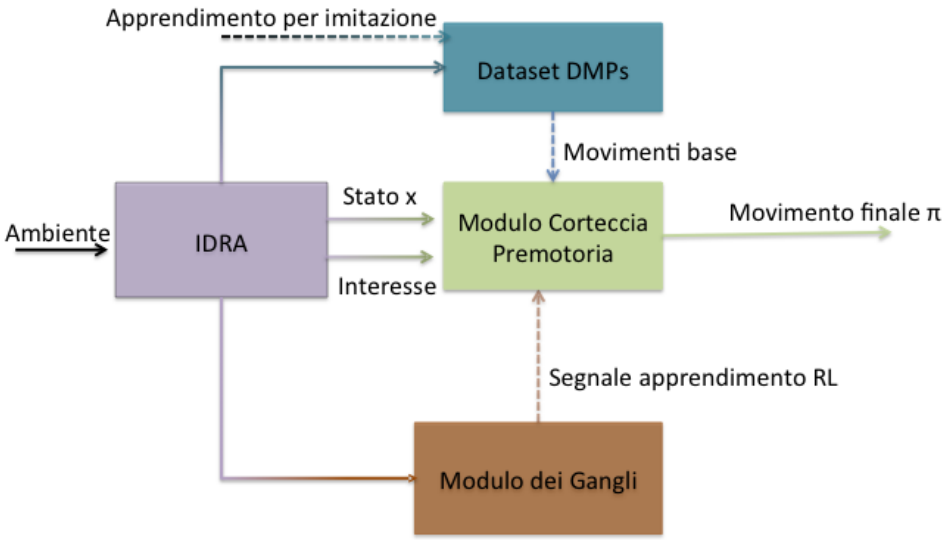

Figura A.1: Schema a blocchi del modello

Il codice seguente rappresenta il punto di entrata del programma e definisce le varie fasi dell'algoritmo.

```
%Mi collego al robot NAO
IP='169.254.95.24';
PORT=9559;


%Definisco i proxy per chiamare le funzioni di Naoqi, il sistema operativo
%del Nao
MOTION=ALMotionProxy(IP, PORT);
PICTURE=ALVideoDeviceProxy(IP,PORT);


%Inizializzo il numero di primitive e i nomi dei DOF del braccio del robot.
NUM_PRIMITIVE=9;
name1='LElbowRoll';
name2='LElbowYaw';
name3='LShoulderPitch';
name4='LShoulderRoll';
NAMES_ARRAY=[name1, name2, name3, name4];
NAMES={name1, name2, name3, name4};




%FASE DI APPRENDIMENTO

%Registro il dataset di DMP
dcp_batch();


%Eseguo l'apprendimento di ICA
%Gli input sono una immagine della camera del nao e una posizione del
%braccio random, concatenati. L'output sono le distanze dell'input
%dalle categorie.
while any(output==0) || isempty(output)
    img=DatasetImmaginiRandom();
    signals(1).sig = img;
    signals(1).filterName = 'LogPolarBW';
```

```
    signals(1).instinct = 'None';
    signals(2).sig = img;
    signals(2).filterName = 'None';
    signals(2).instinct = 'ballPosition';
    n=randi(100,1);
    signals(3).sig = stato(n,:);
    signals(3).filterName = 'None';
    signals(3).instinct = 'None';
    [output, rs] = intentionalArchitecture(signals);
    output;
end



%Definisco i centri delle basis function della policy con media parametrica
for i=1:NUM_PRIMITIVE
    img = CameraNao();
    signals(1).sig = img;
    signals(1).filterName = 'LogPolarBW';
    signals(1).instinct = 'None';
    signals(2).sig = img;
    signals(2).filterName = 'None';
    signals(2).instinct = 'ballPosition';
    n=randi(100,1);
    signals(3).sig = stato(n,:);
    signals(3).filterName = 'None';
    signals(3).instinct = 'None';
    [MU_ACTOR(i,:), rs] = intentionalArchitecture(signals);
end


%Scelgo la prima azione random
InizializzaPesi();


%   FASE APPRENDIMENTO E GENERAZIONE DEL MOVIMENTO
while true

%FASE DI CODIFICA DEGLI STATI E GENERAZIONE INTERESSE
```

```
% IDRA analizza lo stato iniziale
%Definisco l'input di IDRA
img = CameraNao();
signals(1).sig = img;
signals(1).filterName = 'LogPolarBW';
signals(1).instinct = 'None';
signals(2).sig = img;
signals(2).filterName = 'None';
signals(2).instinct = 'ballPosition';
signals(3).sig = stato(r1,:);
signals(3).filterName = 'None';
signals(3).instinct = 'None';
%Prendo la distanza dai cluster
[output, rs] = intentionalArchitecture(signals);
x=output;

  %-------------------------------------------------------------------------
    %   Calcolo la combinazione di primitive.

    %Primo gruppo funzionale di primitive: allungamento
    for i=1:7
        movimento_finale = movimento_finale + (pesi(i)*primitiva(i))/sum(pesi(1:7));
    end

    %Secondo gruppo funzionale di primitive: rotazione
    for i=8:9
       movimento_finale = movimento_finale + (pesi(i)*primimitiva)/sum(pi_pesi(8:9));
    end



    %-------------------------------------------------------------------------
    %   Eseguo l'azione, mi interfaccio con il Nao

    % Controllo che sto generando un movimento ammissibile dal robot
    for i=1:DOF
       movimento_finale(i,find(movimento_finale(i,:)>massimoNao(i)))=massimoNao(i);
       movimento_finale(i,find(movimento_finale(i,:)<minimoNao(i)))=minimoNao(i);
    end
```

```
    esegui_nao(movimento_finale);



    %-------------------------------------------------------------------------
    %   Ricevo/Calcolo il reward
    r=Calcola_reward();


    %-------------------------------------------------------------------------
    %   Chiama Idra che analizza l'input visivo del nao alla fine del movimento.
    %   Restituisce la distanza dell'input corrente dai centri dei cluster
    %   presenti in idra.


    img = CameraNao();
    %Definisco l'input di IDRA
    signals(1).sig = img;
    signals(1).filterName = 'LogPolarBW';
    signals(1).instinct = 'None';
    signals(2).sig = img;
    signals(2).filterName = 'None';
    signals(2).instinct = 'ballPosition';
    signals(3).sig = stato(r1,:);
    signals(3).filterName = 'None';
    signals(3).instinct = 'None';
    %Prendo la distanza dai cluster
    [output, rs] = intentionalArchitecture(signals);
    y=output;


    %-------------------------------------------------------------------------
    %   Esegue una iterazione dell'algoritmo di combinazione di primitive.
    %   w e t sono rispettivamente il vettore dei parametri della policy e
    %   della funzione Q(s,a), z la eligibility trace.
    [w,t,pi_pesi,z]=modulo_gangli(x,y,r,w,t,pesi,z);


end
```

Il seguente codice implementa la funzione CameraNao() che restituisce l'immagine che il NAO vede attraverso la sua camera. In particolare si vuole mostrare come ci si è interfacciati con il robot.

```
function [  ] = CameraNao( )

% Definisco i proxy per interfacciarmi con Naoqi
video = ALVideoDeviceProxy('169.254.95.24',9559);


% Attivo la camera superiore del Nao e imposto la risoluzione
% del video a 120x160
video.subscribeCamera('matlab',int16(1),int16(0),int16(11),int16(30));

% Recupero l'immagine dal NAO
res = video.getImageRemote('matlab');

% Libero la camera
video.unsubscribe('matlab');

% Converto l'immagine in un formato leggibile da matlab,in rgb
rgb  = res{7};
im = uint8(rand(120,160,3));
v = 1;
for j=1:120,
    for i=1:160,
        alpha = bitand(uint8(255),uint8(255));
        alpha = bitshift(uint8(alpha), uint8(24));
        red = bitand(uint8(rgb(i*j)),uint8(255));
        red = bitshift(red,16);

        green = bitand(uint8(rgb(i*j+1)), uint8(255));
        green = bitshift(red,uint8(8));

        blue = bitand(uint8(rgb(i*j+2)), uint8(255));


        red = rgb(v);
        green = rgb(v+1);
        blue = rgb(v+2);
```

```
        im(j,i,1) = red;
        im(j,i,2) = green;
        im(j,i,3) = blue;
        v = v+3;
    end
end


end
```

Questa invece è la funzione per farlo muovere. Abbiamo usato una chiamata alla funzione angleInterpolationWithSpeed del NAO passandogli le posizioni angolari degli giunti del braccio sinistro generate dalla combinazione delle DMP ogni 0.4 secondi.

```
function [] = esegui_nao( angoli_finali)

for i=1:FINE_MOVIMENTO
    angoli=num2cell(angoli_finali);
    MOTION.angleInterpolationWithSpeed(NAMES, angoli, 0.4);
end

end
```

Con l'ultimo estratto di codice vogliamo mostrare una iterazione dell'algoritmo NAC, che è un punto importante del nostro modello.

```
function [ w,t,a1,z,mean_policy ] = Nac( w,t,a,z,x,y,r )
%NAC Algoritmo Natural Actor-Critic, restituisce il vettore dei pesi della
%policy. Ogni chiamata a questa funzione \‘e una iterazione dell’algoritmo.
%   Critico: Sarsa(1)
%   Attore: Policy gaussiana
%
%   Variabili
%   w: vettore colonna dei parametri della policy
%   t: vettore colonna dei parametri della Action-Value function
%   z: vettore colonna delle eligibility traces della Action-Value function
%   a: azione risultante dalla policy
%   r: reward ricevuto
%   y: vettore colonna dello stato dopo l’esecuzione di a.
```

```
%   x: vettore colonna dello stato presente
%   b: learning rate della policy. Deve essere pi\‘u piccolo di quello del
%   critico.

%   Inizializzazioni eseguite nel chiamante alla prima iterazione
%   w=0; t=0; z=0;

b=0.4;
SIGMA = 0.3;

%   Sceglie azione a1 dalla policy. Le azioni ammissibili sono comprese
%   nell'intervallo [0,1].
id=1;
while id==1
    mean_policy=w'*basis_actor(y);
    a1 = mvnrnd(mean_policy,SIGMA);
    if(a1>=0 && a1<=1)
        id=0;
    end
end


%   Valuto la policy corrente aggiornando la Action-value function
[t,z] = sarsa_critic(x,a,r,y,a1,t,z,w);

%   Aggiorno i pesi della policy con la regola del Nac
w = w - b*t; %gradient descent


end
```