

# Equilibri del robot AIBO utilitzant DMPs

Lluís Salord Quetglas  
`l.salord.quetglas@gmail.com`

Escola Tècnica Superior d'Enginyeria Industrial  
Treball Final de Grau  
Grau en Enginyeria en Tecnologies Industrials

17 de juny de 2014



## **Resum**

L'estabilitat del robots...

# Índex

<b>Resum</b>	<b>1</b>
<b>Prefaci</b>	<b>6</b>
Motivació . . . . .	6
Definició del <i>TFG</i> . . . . .	7
Requeriments previs . . . . .	7
<b>1 Introducció</b>	<b>8</b>
1.1 Objectius . . . . .	8
1.2 Planificació . . . . .	9
1.2.1 Planificació inicial . . . . .	9
1.2.2 Planificació final . . . . .	9
1.3 Pressupost . . . . .	9
1.4 Metodologia . . . . .	9
1.5 Abast del treball . . . . .	9
1.6 Estat de l'art . . . . .	9
1.6.1 Robot AIBO . . . . .	10
1.6.2 <i>ROS</i> . . . . .	12
1.6.3 Robots . . . . .	16
1.6.4 Modelat de robots . . . . .	18
1.6.5 Aprenentatge supervisat ( <i>SL</i> ) . . . . .	21
1.6.6 Aprenentatge per reforç ( <i>RL</i> ) . . . . .	23
1.6.7 <i>DMP</i> ( <i>D</i> ynamic <i>M</i> oviment <i>P</i> rimitive) . . . . .	25
1.6.8 <b>P</b> ath <b>I</b> ntegral <b>P</b> olicy <b>I</b> mprovement ( <i>PI</i> <sup>2</sup> ) . . . . .	28
1.6.9 Algorismes avançats . . . . .	30
<b>2 Estudis preliminars</b>	<b>31</b>
2.1 Comunicació mitjançant <i>ROS</i> . . . . .	31
2.1.1 Entre l'AIBO i <i>ROS</i> . . . . .	32
2.1.2 Entre l'algorisme de control i <i>ROS</i> . . . . .	33

2.1.3	Entre l'Arduino i <i>ROS</i> . . . . .	34
2.2	Estabilitat . . . . .	34
2.3	Algorismes de resposta davant pertorbacions . . . . .	35
2.3.1	Model del robot . . . . .	35
2.3.2	Aprenentatge supervisat . . . . .	35
2.3.3	Aprenentatge per reforç . . . . .	36
2.3.4	<i>DMP</i> . . . . .	37
2.3.5	Elecció final . . . . .	39
2.4	Codis amb <i>DMPs</i> implementades . . . . .	39
2.4.1	<i>DMPs</i> de Scott Niekum . . . . .	39
2.4.2	Package complert del robot PR2 del <i>USC-CLMC</i> . . . . .	41
<b>3</b>	<b>Disseny</b>	<b>44</b>
3.1	Model de l'AIBO . . . . .	44
3.2	Moviment del centre de gravetat . . . . .	46
3.2.1	Acceleròmetre MPU6050 i giroscopi GY-521 . . . . .	47
3.2.2	Càlcul de la posició i de la inclinació . . . . .	47
3.3	<i>Goals</i> de les <i>DMPs</i> i funció recompensa . . . . .	49
3.4	Plataforma . . . . .	49
<b>4</b>	<b>Funcionament i execució</b>	<b>50</b>
4.1	Connexions a establir . . . . .	50
4.1.1	Conjunt acceleròmetre-giroscopi . . . . .	50
4.1.2	AIBO a través d' <i>Access Point</i> ( <i>AP</i> ) . . . . .	51
4.2	Processos previs . . . . .	53
4.2.1	Carrega del programa de control del <i>CdG</i> a Arduino . . . . .	53
4.2.2	Execucions per línia de comandes . . . . .	54
4.3	Funcionament de <i>DMPs</i> sense $PI^2$ . . . . .	54
4.4	Funcionament de <i>DMPs</i> amb $PI^2$ . . . . .	54
<b>5</b>	<b>Conclusions</b>	<b>55</b>
	<b>Agraïments</b>	<b>56</b>
	<b>Referències</b>	<b>62</b>
	<b>Annexos</b>	<b>63</b>
<b>A</b>	<b>Instal·lació de <i>ROS</i>, llibreries d'Urbi i paquet <i>aibo_server</i></b>	<b>64</b>
<b>B</b>	<b>Bibliografia</b>	<b>66</b>

# Índex de figures

6.1	AIBO ERS-7 (vista frontal) [44] . . . . .	11
6.2	AIBO ERS-7 (vista posterior) [44] . . . . .	12
6.3	AIBO ERS-7 (vista inferior) [44] . . . . .	12
6.4	REEM-C [47] . . . . .	17
6.5	BigDog [43] . . . . .	17
6.6	LittleDog [21] . . . . .	18
6.7	Esquema de cada arquitectura d'aprenentatge [36] . . . . .	21
6.8	Gràfics explicatius de la funció no lineal $f(s)$ . . . . .	27
6.9	Efecte dels pesos $w_i$ (representats per $\theta$ ) i de l'objectiu ( <i>goal</i> ) . . . . .	27
6.10	Comparació de l'algorisme de <i>DMP</i> original (esquerra) i el modificat per [39] (dreta) . . . . .	28
6.11	Algorisme genèric dels mètodes de millora de política [55] . . . . .	29
1.1	Comunicació entre l'AIBO, Urbi i <i>ROS</i> . . . . .	33
3.1	<i>DMP</i> a partir una trajectòria de demostració . . . . .	38
4.1	Diagrama fluxos del codi de Scott Niekum . . . . .	41
4.2	Procés iteratiu del codi <i>Policy_learning</i> . . . . .	43
1.1	Mesures en <i>mm</i> de l'AIBO [5] . . . . .	45
1.2	Models de l'AIBO en l'entorn <i>rviz</i> . . . . .	46
1.1	Esquemes del funcionament del bus I <sup>2</sup> C d'escriptura (dalt) i de lectura (baix). . . . .	51
1.2	Connexions del conjunt acceleròmetre-giroscopi amb l'Arduino. . . . .	52
1.3	Esquema de connexions entre AIBO- <i>ROS</i> i sensor-Arduino- <i>ROS</i> . . . . .	53

# Índex de taules

6.1	Rangs de funcionament de les articulacions [13] . . . . .	13
6.2	Relació tipus de model amb arquitectura d'aprenentatge [36] . . . . .	20

# Prefaci

## Motivació

A moltes persones que tenen devoció pel món de la robòtica, aquesta els hi ha vingut des de ben joves; aquest no és el meu cas. A mi m'agradava la informàtica. Essent un escolar vaig aprendre de forma autodidacta a programar en *C++* i crear un servidor propi. Fet que ajudà, en arribar a la *UPC*, a divertir-me amb assignatures relacionades amb programació, però em faltava veure reflectit el meu treball amb alguna utilitat física. No va ser fins que a l'assignatura de "Projectes II" vam controlar un mecanisme a través d'un microcontrolador *Arduino*<sup>1</sup>, amb programació basada en *Wiring* [4], que vaig veure clar cap on volia enfocar el meu futur, la robòtica.

D'aquesta motivació, que ha crescut poc a poc, n'ha sorgit el perquè de desenvolupar aquest *TFG* (**T**reball **F**inal de **G**rau). A més, a cada pas que he fet en el treball, com més he vist i entès la gran complexitat d'altres robots, com el *Big Dog* de *Boston Dynamics* o el *NAO* d'*Aldebaran Robotics*<sup>2</sup>, entre d'altres, més estímuls tenia per avançar i millorar.

Ara bé, deixant de banda les pròpies ganes de treballar en robòtica, també he tingut en compte la preparació pel futur professional. Per això, en molts casos, a l'hora de fer alguna elecció, he intentat escollir la que fos més innovadora i útil en un futur proper. Aquest fet es visualitza tant en l'elecció dels llenguatges de programació utilitzats, com també en l'algorisme de resolució de la problemàtica de treball en qüestió. Per tant, queda palès que aquest treball té com a motivació la combinació d'entusiasme pel món de la robòtica, la millora continuada d'un mateix per tal de poder arribar a un bon futur professional en aquest camp i l'objectiu d'integrar els coneixements i habilitats apresos al llarg del grau.

---

<sup>1</sup>*Arduino* és la marca d'una família de microcontroladors.

<sup>2</sup>En l'estat de l'art es dona una breu explicació d'aquests i altres robots.



## Definició del *TFG*

### Requeriments previs

Per fer el seguiment del *TFG* que es presenta, tot i que s'ha intentat donar explicació als conceptes que podrien ser més complicats d'entendre, és necessari disposar d'uns certs coneixements previs o nocions en *ROS* (per **R**obot **O**perative **S**ystem)<sup>3</sup>, Control Automàtic, llenguatge de programació *C++* i *Python*, Arduino, Mecànica i Xarxes de dades.

A banda del que s'ha esmentat anteriorment, per reproduir de nou l'experiència és molt recomanable llegir articles sobre aprenentatge de robots, tant per reforç, com utilitzant *DMPs* (per **D**ynamic **M**ovement **P**rimitives). Algunes de les referències més recomanades es poden trobar a l'apartat de Referències o Bibliografia.

Finalment, el més necessari de tot és tenir molta motivació i paciència, per així, no decaure davant les adversitats que un s'arriba a trobar i continuar endavant.

---

<sup>3</sup>S'explica detalladament en l'apartat 1.6.2.

# Capítol 1

## Introducció

Els robots són mecanismes programables i accionats per dos o més eixos amb cert grau d'autonomia, movent-se en el seu entorn, per realitzar les tasques previstes [18]. Actualment, els robots s'utilitzen principalment a nivell industrial, per la realització d'accions de forma més exacte i barata o per treballs perillosos o repetitius. Ara bé, també existeix el cas de robots com l'AIBO, de Sony, entre d'altres robots, que poden ser utilitzats tant per entreteniment de l'usuari, com robot social o per la investigació o millora dels robots actuals.

Els estudis en robòtica es poden centrar tant en el *hardware*, com en el *software*. Tot i només enfocar-se en una de les dues branques, sempre es requereix de l'altre en més o menys proporció. El fet és que el dissenyar i fabricar el *hardware* necessari s'emporta una gran partida del pressupost. Per això, en les investigacions que no compten amb grans pressuposts, com ara estudis universitaris, és comú l'ús de robots comercials en que es pot modificar el codi intern, com és el cas de l'AIBO.

A banda d'aquest robot, n'hi ha molts més que són utilitzats per dur a terme investigacions, tant robots comercials, com dissenyats i fabricats des de zero. El treball present es centra en l'AIBO, l'estabilitat, el modelat i l'aprenentatge d'un robot, per tant, tan sols es fa l'estudi d'antecedents d'aquests casos.

### 1.1 Objectius

L'objectiu principal del present *TFG* és l'optimització de l'adaptabilitat d'un robot quadrúpede, en aquest cas l'AIBO, a plans inclinats desconeguts pel robot. Per arribar a aquest objectiu s'han hagut de marcar uns objectius més concrets:

- Dissenyar un entorn de treball complet que permeti que l'algorisme utilitzat pugui ser processat en l'ordinador i enviar la informació necessària de forma remota a l'AIBO. En aquest cas l'entorn de treball tal que permeti això és el *ROS*.
- Utilització de l'algorisme més adequat, tenint en compte tant l'entorn del robot i ell mateix, com l'abast del treball. Per això s'haurà de fer un estudi dels diferents mètodes existents que podrien ser útils per l'objecte del treball.
- Dur a terme una fase d'aprenentatge pel robot. Per poder fer-ho, abans, s'haurà d'haver fet un estudi en profunditat del aprenentatge per reforç.
- Realització de diferents proves per comprovar el correcte funcionament. Tant per poder comprovar, com per fer la fase d'aprenentatge del robot, és necessita d'una plataforma mòbil que en aquest cas ja està construïda, pel Carlos Ramos (estudiant de l'*EPSEVG*) [45], però s'ha de millorar per fer-la més robusta.

## 1.2 Planificació

### 1.2.1 Planificació inicial

### 1.2.2 Planificació final

## 1.3 Pressupost

## 1.4 Metodologia

## 1.5 Abast del treball

## 1.6 Estat de l'art

En la branca d'investigació sobre l'estabilitat en robots, tant bípedes, com quadrúpedes, hi ha multitud de tesis, treballs, articles, etc. Tots ells, centrant-se en un o altre aspecte com són: el punt de moment zero (*ZMP*, per *Zero Moment Point*), modelat de robots, aprenentatge supervisat, per reforç o *DMP*, generador de patrons centrals (*CPG*, per

*Central Pattern Generator*), algorismes genètics (*GA*, er *Genetic Algorithms*)<sup>1</sup>, i molts altres.

Tot seguit, s'exposa un conjunt d'antecedents, organitzat en diferents àmbits, importants tots ells tant per realitzar l'experiència, com per entendre els factors que han conduït a cadascuna de les decisions preses.

### 1.6.1 Robot AIBO

L'AIBO (*Artificial Intelligence RoBot*) és un robot quadrúpede dissenyat i fabricat per Sony Corporation, amb aparença canina. El primer model, que va ser tret al mercat, fou el *ERS-110* l'any 1999, a partir d'aquest, i després de tres generacions, el 2003 s'arribà al *ERS-7*, molt més sofisticat que els seus predecessors, tot i que en el 2006 s'aturà la producció de la família AIBO. L'*ERS-7* és el model que s'estudia i s'utilitza en el present treball.

Aquest és considerat un robot autònom, per tant, és capaç d'extreure informació del seu entorn, funcionar per un període llarg sense la intervenció humana, moure alguna o totes les parts d'ell mateix dins d'un entorn de treball sense l'ajut d'un humà i, finalment, evitar situacions de perill per les persones, els bens o ell mateix, si no és per especificacions del propi disseny.

L'aplicació d'aquest robot autònom està enfocada en ser utilitzat en propòsits d'entreteniment, tot i ser, en molts casos, utilitzat en tasques d'investigació. Els robots autònoms corrents tendeixen a ser dissenyats per desenvolupar tasques de seguretat o treballs peril·losos, ara bé, en aquests casos no es pot tolerar cap tipus d'error en les operacions crítiques. Mentre que si estan dissenyats per usos d'entreteniment, en el cas que es produís algun error no seria un amenaça per la vida [11].

Els dissenyadors de l'AIBO han perseguit l'objectiu d'aconseguir que el comportament sigui el més realista possible, que sembli viu. Per assolir-ho, han avançat per diferents camins:

- Estímuls
  - Comportaments reflexius i deliberats segons una escala de temps.
  - Comportaments per ordres externes i per desigs interns (instints i emocions).
  - Motivacions independents donades per parts del robot com coll, cua i potes.

---

<sup>1</sup>La majoria d'aquests conceptes seran explicats al llarg d'aquest apartat.

- Instints i emocions amb els que pot canviar el comportament davant d'altres estímuls externs.
- Aprenentatge i evolució, inicialment és com un nadó sense pràcticament cap coneixements. Així com passa el temps, l'AIBO aprèn i creix segons com el tractis. Per tant, podria arribar a comportar-se com un noi entremaliat, si no se li dona l'atenció necessària.

## Hardware

Les característiques del robot són les següents: [3]

- Processador MIPS R7000 de 576 MHz
- Memòria *RAM* de 64 MB
- *LAN* sense fils, 802.11b (estàndard)
- Targeta interna de memòria lectura/escriptura
- 18 articulacions *PID*, cadascuna amb un sensor de força
  - 4 potes
    - \* 3 articulacions cadascuna (elevació, rotació i genoll)
    - \* 1 sensor de pressió a cada peu
  - 3 articulacions al coll (moviment horitzontal, vertical i inclinació)
  - 2 articulacions a la cua (moviment vertical i inclinació)
  - 1 articulació a la boca

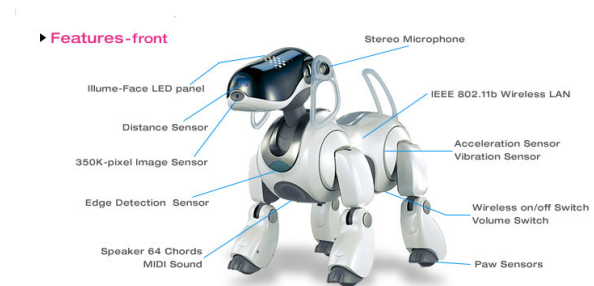


Figura 6.1: AIBO ERS-7 (vista frontal) [44]

- 2 orelles, on hi ha els micròfon estèreo i amb una articulació booleana (posició dalt o baix)
- Altaveus de 500 mW
- 26 LEDs independents
- Càmera de vídeo
  - Sensor d'imatge CMOS
  - 56.9° ample i 45.2°
  - Resolucions:  $208 \times 160, 104 \times 80, 52 \times 40$
  - 30 imatges per segon
- 3 sensors de distància per infrarojos (un al cos i dos al nas, d'aquests dos, un és per objectes llunyans i un altre per pròxims)
- Acceleròmetres  $X$ ,  $Y$  i  $Z$
- 4 botons sensorials de pressió (un al cap i tres al llom)
- 1 botó booleà sota la boca
- Sensor de vibració
- Actualització dels sensors cada 32 ms, amb 4 mostres per actualització
- Dimensions:  $319 \times 180 \times 278$
- Pes aproximat: 1,65 kg (bateria i targeta de memòria incloses)

► Features-back



Figura 6.2: AIBO ERS-7 (vista posterior) [44]

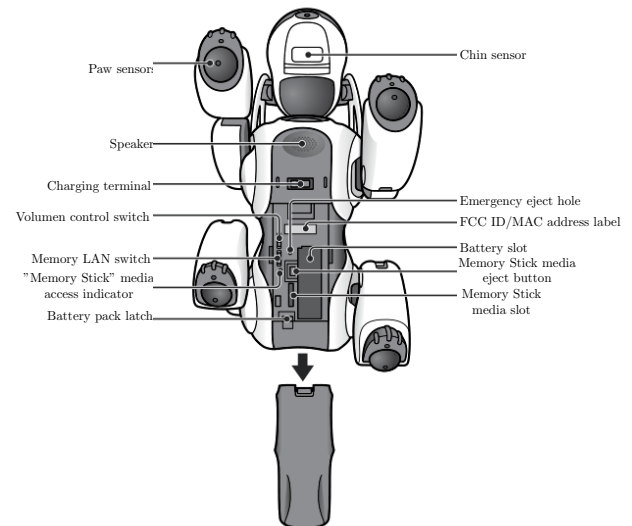


Figura 6.3: AIBO ERS-7 (vista inferior) [44]

Les articulacions *PID*, segons la seva funció i les pròpies limitacions físiques tenen uns rangs de treball diferents, aquest són els que s'exposen tot seguit:

## Software

### 1.6.2 ROS

*Robot Operating System* [46] és un entorn de treball *open-source* i flexible per la programació de robots. Les bases d'aquest projecte s'iniciaren en unes investigacions a Stanford el 2007, on es varen dur a terme diferents prototips d'entorns de treball per programari de robots, com ara **ST**anford **A**rtificial **I**ntelligence **R**obot (*STAIR*) o **P**ersonal **R**obotics (*PR*). Més endavant, *Willow Garage*, una empresa inversora en robòtica, va proveir re-

Name	Range	Units	Description
legRF1	range=[-134.000000,120.000000]	unit=deg	Right fore legJ1
legRF2	range=[-9.000000,91.000000]	unit=deg	Right fore legJ2
legRF3	range=[-29.000000,119.000000]	unit=deg	Right fore legJ3
legRH1	range=[-134.000000,120.000000]	unit=deg	Right hind legJ1
legRH2	range=[-9.000000,91.000000]	unit=deg	Right hind legJ2
legRH3	range=[-29.000000,119.000000]	unit=deg	Right hind legJ3
legLF1	range=[-120.000000,134.000000]	unit=deg	Left fore legJ1
legLF2	range=[-9.000000,91.000000]	unit=deg	Left fore legJ2
legLF3	range=[-29.000000,119.000000]	unit=deg	Left fore legJ3
legLH1	range=[-120.000000,134.000000]	unit=deg	Left hind legJ1
legLH2	range=[-9.000000,91.000000]	unit=deg	Left hind legJ2
legLH3	range=[-29.000000,119.000000]	unit=deg	Left hind legJ3
neck	range=[-79.000000,2.000000]	unit=deg	Neck tilt1
headTilt	range=[-16.000000,44.000000]	unit=deg	Neck tilt2
headPan	range=[-91.000000,91.000000]	unit=deg	Head pan
tailPan	range=[-59.000000,59.000000]	unit=deg	Tail pan
tailTilt	range=[2.000000,63.000000]	unit=deg	Tail tilt
mouth	range=[-58.000000,-3.000000]	unit=deg	Mouth

Taula 6.1: Rangs de funcionament de les articulacions [13]

cursos per tal de millorar el concepte i permetre crear implementacions correctament testejades. Finalment, amb la col·laboració desinteressada d'incomptables investigadors, millorant el nucli de *ROS* i les eines principals que proveeix, s'ha arribat al que és ara, una plataforma àmpliament utilitzada en les investigacions de robòtica.

En el moment de la redacció d'aquest treball, la versió més actual de *ROS* és la *ROS Hydro Medusa*, publicada el setembre de 2013, i pròximament es publicarà la *ROS Indigo Igloo*. La Hydro està dissenyada especialment per Ubuntu 12.04 LTS (*Precise*), tot i suportà també altres sistemes Linux, Mac OS X, Android i Windows en altres graus.

## Estructura de *ROS*

*ROS* ofereix una interfície que permet la comunicació entre processos per tal de processar dades conjuntament, és comú referir-s'hi com a capa intermèdia. Els conceptes fonamentals de la implementació de *ROS* són els **nodes**, **Master**, **messages**, **services**, **topics** i **bags**.

- **Nodes:** Els **nodes** són processos que realitzen càlculs. Típicament, un sistema compren multitud de **nodes**. En aquests casos és útil entendre les comunicacions entre **nodes** com un graf, amb arcs que uneixen els que s'estan comunicant.

- **Master:** El *ROS Master* proveeix els noms d'enregistrament dels **nodes**, **topics** i **services** existents als altres **nodes**. Per tant, el **Master** rep la informació de registre dels **nodes** i després aquest informa als altres **nodes** per tal que puguin establir, entre ells, connexions de forma adequada.
- **Messages:** Els **nodes** es comuniquen un amb l'altre mitjançant **messages**. Aquest són simplement estructures de dades, que poden anar des de tipus **integer**, **float**, **boolean** fins a **array**.
- **Topics:** Un **node** envia un **message** mitjançant la publicació d'aquest en un **topic** donat. El **topic** és el nom que s'utilitza per identificar el contingut d'un **message** concret.
- **Services:** El **service** és el nom que ha d'utilitzar un **node** per enviar un **message**, amb la funció de sol·licitar una resposta que depèn del **message** enviat.
- **Bags:** Els **bags** són un format per guardar i poder reproduir un altre cop les dades de **messages** de *ROS*. Aquests són de gran importància a l'hora d'emmagatzemar dades i, per tant, per desenvolupar i testejar algorismes.

Aquesta capa intermèdia ofereix dos models de comunicació: (1) sistema de *publicació/subscripció* (**publisher/subscriber**) ; i (2) utilitzant **services**.

1. El sistema de *publicació/subscripció* és anònim, asíncron i les dades poden ser capturades i rellegides sense canvis en el codi. Per tant, si per fer una certa tasca es requereix de les dades d'una altra tasca, com per exemple un sensor, llavors a partir de subscriure's al **topic** corresponent es poden llegir les dades que publica la tasca (sensor). Pot haver-hi múltiples publicadors i subscriptors per un únic **topic** i, en general, entre ells no saben de l'existència dels altres.
2. Els **services** estan definits per dos **messages**, un és la demanda que ha fet el **node** i l'altre és la resposta a aquesta demanda. Per tant, el seu ús és molt simple, en el moment que es crida un **service**, amb les dades que aquest requereixi, el procés dona una resposta al **node** segons les dades que s'han enviat.

## Objectius de *ROS*

El principal objectiu de *ROS* és poder *reutilitzar* el codi de desenvolupament i d'investigacions en robòtica. L'estructura de processos distribuïts permet aquest fet, ja que pot executar-se un procés (amb un codi determinat) de forma individual i acoblar-se fàcilment al conjunt. A més, aquests processos poden agrupar-se en **Packages** i **Stacks**, per ser



compartits de forma senzilla.

D'altra banda, també es tenen altres finalitats [42]: (1) descentralització; (2) plurilingüisme; (3) estar basat en eines; (4) ser una capa intermèdia fina; (5) gratuïta i *open-source*.

### 1. Descentralització

*ROS* està estructurat de forma que els processos estan distribuïts, amb la possibilitat de trobar-se en *hosts* diferents, però funcionant conjuntament. Altres entorns de treball, que poden també treballar amb múltiples processos i *hosts*, si es basen en un servidor central, podrien tenir problemes en una xarxa heterogènia<sup>2</sup>.

### 2. Plurilingüisme

Cada programador és un món, cadascú té el seu llenguatge de programació preferit, sigui per la raó que sigui. Per això, *ROS* s'ha dissenyat per ser un llenguatge neutral. Actualment, *ROS* admet quatre llenguatges de programació: (1) *C++*, (2) *Python*, (3) *Octave* i (4) *LISP*, havent altres en desenvolupament.

### 3. Basat en eines

S'ha optat per dissenyar un nucli simple, on s'utilitzen multitud d'eines per construir i fer funcionar els diversos components de *ROS*, en lloc, de dissenyar un enorme entorn de treball, tot en un. Tot i haver-se implementat alguns serveis en el propi nucli, s'ha intentat distribuir tot en mòduls separats. La pèrdua d'eficiència compensa els guanys en estabilitat i complexitat del conjunt.

### 4. Capa intermèdia fina

En molts casos, és molt difícil “extreure” la funcionalitat d'un codi, del seu context original, per a poder ser reutilitzat, això és degut a factors provocats pel propi entorn de treball d'origen. Per això, en *ROS* s'indueix a la independència dels algorismes, amb el nucli del *ROS*, creant-los en llibreries separades. Es facilita l'extracció de codi i la seva reutilització a través d'aquest fet, entre d'altres característiques de la interfície.

### 5. Gratuït i *open-source*

El codi natiu de *ROS* està disponible públicament. Aquest és un fet que permet facilitar el testeig i correcció de *software* en tots els nivells.

---

<sup>2</sup>Una xarxa heterogènia és una xarxa de connexió d'ordinadors i altres dispositius amb diferents sistemes operatius i/o protocols [9].

## Eines de *ROS*

Com s’ha comentat breument en l’apartat anterior, *ROS* és basa, en gran part, en la multitud d’eines que disposa. Aquestes eines poden arribar a dur a terme diverses tasques diferents, per exemple, navegar per l’arbre de codi font, obtenir i establir els paràmetres de configuració, visualitzar les connexions entre processos, mesurar la utilització d’ample de banda, exposar de forma gràfica les dades dels `message`, i més. A continuació es comenten breument alguns dels més utilitzats:

- **rviz**

**rviz** és un entorn de visualització 3D que pot combinar les dades dels sensors del robot i el model que es té, juntament amb altres dades 3D que se li aporti, per poder visualitzar el conjunt.

- **rosviz i rxbag**

**rosviz** és la comanda que et permet emmagatzemar i reproduir de nou les dades d’un `message` en un arxiu `bag`. Per altra banda, **rxbag** és un visualitzador per a les dades emmagatzemades dins els arxius `bag`.

- **rxplot**

**rxplot** permet veure dades escalars publicades en els `topics` de *ROS*.

- **rxgraph**

**rxgraph** exposa visualment amb un gràfic com funcionen els processos de *ROS* i les seves connexions, en aquell instant.

### 1.6.3 Robots

Alguns dels robots sobre els que s’hi ha investigat, amb temàtiques relacionades amb el treball present són:

**QRIO** Robot humanoide dissenyat i fabricat per Sony Corporation, és el successor de l’AIBO. Entre d’altres articles i investigacions que se n’ha fet es troba [35] sobre l’estabilitat d’un robot bípede a l’hora de caminar, córrer i saltar. Basat en la teoria del *ZMP*.

**REEM-C**

Aquest humanoide és el creat per PAL Robotics [47]. Destaca pel fet de ser el primer bípede enfocat en la investigació i basat 100% en *ROS*. El REEM-C està basat, entre d'altres teories, en el *ZMP* i en l'aprenentatge propi del robot. A més, les seves característiques de reconeixement de veu, manipulació d'objectes i d'interacció amb humans, és una eina educativa molt útil [47].



Figura 6.4: REEM-C [47]

**ASIMO** És un altre robot humanoide, aquest desenvolupat per HONDA, a partir de l'any 2000. Inicialment, en els seus predecessors, tan sols s'havia plantejat el fet de crear un robot mòbil bípede, però, poc a poc, s'hi han incorporat més facultats, fins arribar a ser un dels humanoides amb els moviments més semblants al dels humans [16]. De les referències llegides en el moment de redactar el treball, de l'ASIMO hi ha estudis sobre la planificació dels passos per tal d'evitar obstacles [7] i sobre la interacció amb els humans [34].

**BigDog**

És un dels grans robots que s'han creat a l'empresa *Boston Dynamics*, prenent el que va ser inicialment desenvolupat en la *DARPA* [43]. Aquest “gos” va ser dissenyat per ús militar, en concret, per acompanyar als soldats portant la càrrega necessària en terrenys on no podria desplaçar-se un vehicle convencional. Aquest quadrúpede és dinàmicament estable<sup>a</sup> gràcies al gran conjunt de sensors i actuadors que arriba a tenir. A banda d'un sistema mecànic molt complert, també s'hi ha implementat algorismes d'aprenentatge per reforç (**Reinforcement Learning**<sup>b</sup>), en concret *DMP* (*Dynamic Movement Primitives*)<sup>c</sup> [43].



Figura 6.5: BigDog [43]

<sup>a</sup>Sistema que és estable tenint en compte els efectes inercials i altres components dinàmiques que apareixen en el propi sistema [41].

<sup>b</sup>Aprenentatge per reforç s'explica en detall en l'apartat 1.6.6. En molts casos es abreviat com *RL*.

<sup>c</sup>*DMP* (*Dynamic Movement Primitives*) s'explica en detall en l'estat de l'art a l'apartat 1.6.7.

**LittleDog**

Aquest quadrúpede és el predecessor del BigDog. Té la mateixa base que l'anterior, tot i que en aquest és on s'ha fet més estudi del aprenentatge del robot. La investigació que s'hi ha fet al damunt, tant d'aprenentatge, com de criteri de *ZMP* és pot entendre de forma genèrica en [21].



Figura 6.6: LittleDog

A banda dels nombrats anteriorment, existeixen molts altres robots amb potes que han servit per aprofundir en coneixements diversos, com l'estabilitat o l'aprenentatge dels robots. Molts d'ells han sigut creats des de zero, com són els següents exemples: (1) el PLEO, un robot “*dinosaure*”, que en el projecte [32] se li aporta una millora substancial en la comunicació robot-ordinador; (2) el BISAM, on en l'article [1] s'estudia com provocar que els moviments siguin més semblants als d'un mamífer quadrúpede; (3) el MRWALL-SPECT IV, on l'autor d'aquests articles [29] i [30] es centra en l'adaptabilitat del quadrúpede a diferents terrenys; (4) el MERO, estudiat en [19] per fer una anàlisi d'estabilitat quan aquest es desplaça; (5) per últim, també hi ha els casos d'hexàpodes, tant per l'estudi del caminar amb el criteri de les tres potes [28], com en la construcció des de zero [31], entre d'altres.

### 1.6.4 Modelat de robots

Un model d'un robot és un sistema virtual que representa de forma aproximada la cinemàtica i/o la dinàmica d'un robot, mitjançant formes geomètriques enllaçades entre elles amb una configuració determinada. Aquesta és la base d'un model, ara bé, se li poden afegir complements, com un aspecte visual més vistós, amb alguna textura o concretar quins són els actuadors o sensors, on situar-los, etc.

Durant molt temps, per utilitzar un robot es requeria d'un model. D'aquesta manera, l'autòmat podia saber en quina posició es trobava, en tot moment, i reaccionar de forma correcte. Si no es feia seguint aquest procediment, l'única opció era que el programador tingués en compte totes les diferents possibilitats de fallada i les corregís, sent aquesta un tasca molt complicada.

### Classificació de models

Per crear el model d'un robot existeixen diverses possibilitats. La més rudimentària és prenent les mesures del propi robot i introduir-les al programa, avui dia aquest mètode és poc utilitzat quan es vol un model molt acurat. El més típic, en aquests casos, és utilitzar

el propi robot, amb una arquitectura d'aprenentatge òptima, per fer el model. Aquesta arquitectura es basa en un sistema realimentat amb (1) robot, (2) el model en construcció i (3) un controlador per la realimentació; per així arribar finalment a desenvolupar el model, està il·lustrat molt clarament a la Figura 6.7.

Ara bé, existeixen tants tipus diferents de models, com també formes diferents de crear-los segons [36]:

- Tipus de models:

**Directes** Aquest preveu el pròxim estat d'un sistema dinàmic, donada un acció i estat actual. Per tant, els models directes representen la relació causal entre estats i accions. Una de les seves utilitats és en el control automàtic clàssic, entre d'altres.

**Indirectes** Per altra banda, aquests preveuen l'acció requerida pel sistema per passar d'un estat actual al desitjat pel futur. A diferència dels directes, aquest representen una relació anticausal. Aquest és molt utilitzat en estudis de dinàmica inversa, ja que la relació inversa està ben definida.

**Mixtes** La combinació dels dos models dona el model mixt. La idea és que la informació del model directe pugui ajudar en la manca d'unicitat del model indirecte, ja que el model indirecte té infinitat de solucions.

**De predicció de múltiples passos** Finalment, aquest és principalment utilitzat per la predicció d'una acció o estat futur concret, sense la disponibilitat de les mesures en del moment en qüestió.

Cadascun dels models té unes característiques que el defineixen, però aquestes delimiten els diferents modes d'aprenentatge que poden ser utilitzats per crear-los. Per tant, no tots els models poden ser creats a partir de qualsevol arquitectura d'aprenentatge. Aquest fet s'exemplifica a la Taula 6.2.

- Arquitectura d'aprenentatge, figura 6.7:

**Modelat directe** El model s'extreu amb l'aprenentatge a partir de l'observació dels *inputs* i els *outputs* del propi robot. Aquesta és probablement la tècnica d'aprenentatge més freqüent per aproximació de models.

**Modelat indirecte** Una de les tècniques per dur a terme modelat indirecte és l'aprenentatge de l'error de realimentació. Aquest utilitza l'error creat pel controlador de realimentació per tal d'aprendre i crear així el model.

**Aprenentatge amb professor distal** La idea és crear un model invers, però guiat

Model Type	Learning Architecture
Forward Model	Direct Modeling
Inverse Model	Direct Modeling Indirect Modeling
Mixed Model	Direct Modeling (if invertible) Indirect Modeling Distal-Teacher
Multi-step Prediction Model	Direct Modeling

Taula 6.2: Relació tipus de model amb arquitectura d'aprenentatge [36]

amb un model directe, per tal de minimitzar la manca d'unicitat del model invers.

## Simulació amb models

Un dels beneficis de tenir el model d'un robot és poder fer simulacions virtuals del robot, de tal manera que no es provoca cap desgast al robot real, ni es poden donar situacions de perill. Tot i ser de gran utilitat, els simuladors també tenen les seves limitacions, és difícil simular la física d'un robot (actuadors, interaccions amb l'entorn, sensors...) de manera realista. A més, passar de simulacions a un robot real no sempre és fàcil [15].

Ara bé, existeixen una gran multitud de simuladors cada un amb les seves peculiaritats. Alguns dels que s'ha pogut extreure informació i que podrien ser de més interès són els següents:

**Webots<sup>TM</sup>** [33] Simulador de robots mòbils desenvolupat per Cyberbotics Ltd. La física està basada en Open Dynamic Engine (*ODE*), per així simular una dinàmica més acurada. Aquest *software* proveeix un entorn de treball per modelar i programar el teu propi robot, a més inclou models de diversos robots com són Sony AIBO, Khepera, Lego Mindstorms<sup>TM</sup> o Pioneer2. Però té la desventatge que és un simulador de pagament.

**SimRobot** [25] Aquest és un simulador genèric de robots en 3D. Com el Webots<sup>TM</sup>, el SimRobot també es basa en la física d'*ODE*. Un dels inconvenients d'aquest simulador és que no es possible transferir els controladors de la simulació al robot real.

**Gazebo** [22] És un simulador multi-robots en 3D. Aquest, al igual que Webots<sup>TM</sup>, permet el modelat del teu propi robot, tot i ser en llenguatge C. També es diferencia pels

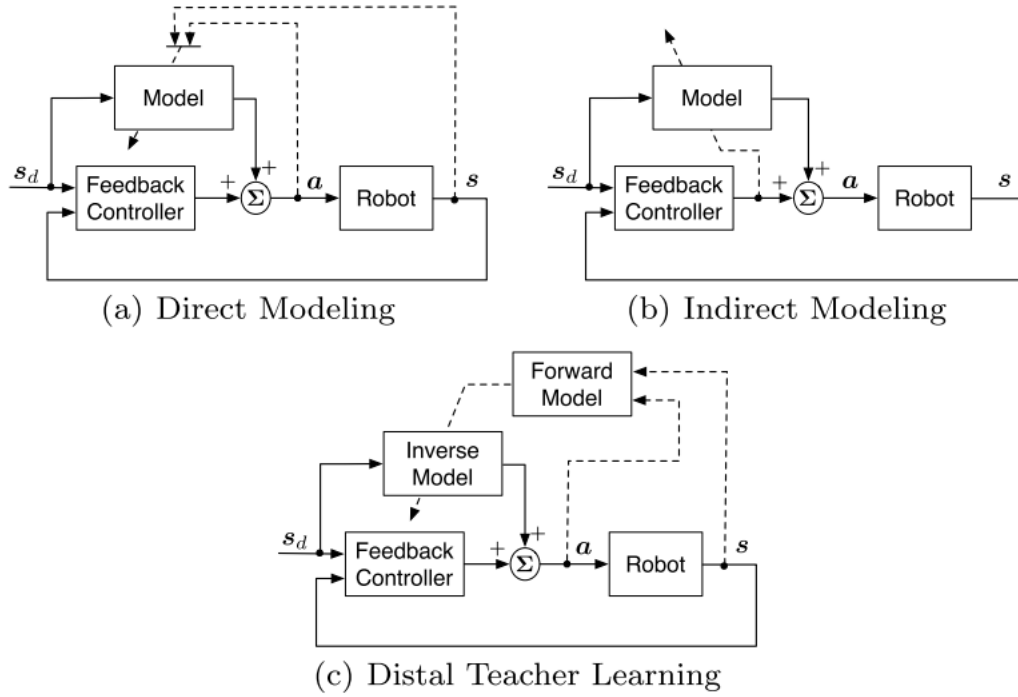


Figura 6.7: Esquema de cada arquitectura d'aprenentatge [36]

models de robot que inclou, que són el Pioneer2DX i el SegwayRMP.

Com s'ha mencionat, Webots<sup>TM</sup> inclou un model del Sony AIBO, dissenyat en [15]. Aquest té implementat l'estructura cinemàtica, propietats dinàmiques<sup>3</sup>, el seu control i l'aspecte gràfic. Per altra banda, també es poden simular els sensors de distància i els de les potes. El model té certes limitacions, els sensors del llom, cap, acceleròmetres i tèrmics no estan implementats per poder ser simulats.

### 1.6.5 Aprenentatge supervisat (SL)

En l'aprenentatge supervisat (en estadística anomenat *anàlisi clúster*), un agent extern presenta una sèrie de dades d'exemple o d'entrenament, que són prediccions correctes a fer en diferents situacions [23]. A partir d'aquestes dades d'entrenament, s'ha d'extreure un model estadístic per tal que, en una situació desconeguda, s'esculli l'acció correcta. L'aprenentatge supervisat és, segons [8], la metodologia més important d'aprenentatge automàtic i amb molt pes en el processament de dades multimèdia.

Les dades a estimar poden ser binàries, on s'escull si una dada desconeguda és d'un tipus (p. e. pertany a un grup o no), o numèriques, on s'utilitza la regressió per aproximar. Tant

<sup>3</sup>Masses i moments d'inèrcia.

siguin unes o altres, les bases de l'aprenentatge supervisat són: (1) el model estadístic, (2) la funció de pèrdua i la d'error d'aproximació, i (3) procediment d'optimització [2].

1. El model es representa com  $g(x|\theta)$ <sup>4</sup>, on  $g(\cdot)$  és la classe d'hipòtesi i els valors de  $\theta$  donen una hipòtesi en concret, d'entre les possibles en el model.
2. La funció de pèrdua,  $L(\cdot)$ , quantifica la diferència entre la sortida desitjada,  $r^t$ , i l'aproximació  $g(x^t|\theta)$ , mentre la suma de les pèrdues de cada cas és l'error d'aproximació

$$E(\theta|X) = \sum_t L(r^t, g(x^t|\theta)) \quad (6.1)$$

3. El procediment d'optimització per trobar  $\theta^*$  que minimitza l'error total,  $E(\theta|X)$ , és:

$$\theta^* = \arg \min_{\theta} E(\theta|X) \quad (6.2)$$

En models complexos, seria més convenient utilitzar mètodes basats en el gradient (p. e. gradient descendent, gradient conjugat, gradient biconjugat...) o l'algorisme de recuita simulada<sup>5</sup>.

Un dels algorismes més simples és la classificació per veí més proper, aquest és molt útil per entendre el funcionament bàsic de l'aprenentatge supervisat [27]. En aquest cas, les dades d'entrenament estan etiquetades, per tant, cada una pertany a un grup en concret. Suposem que es té alguna forma de fer el càlcul de la distància entre dues mostres  $x_1$  i  $x_2$ , expressat com  $D(x_1, x_2)$ .

Llavors amb la forma simplificada, pel cas de binàries, de (6.2)

$$i^* = \arg \min_{i \in \{1 \dots n\}} D(x_t, x_i) \quad (6.3)$$

Sent  $x_t$  la dada a classificar i  $x_i$  l'exemple més pròxim. Després de trobar  $i^*$ , s'assigna l'etiqueta de  $x_i$  a  $x_t$ , queda així classificada la dada. Per suposat, aquesta assignació és una suposició, pot ser correcte o incorrecte.

<sup>4</sup>La  $x$  són les entrades, mentre  $\theta$  són els paràmetres.

<sup>5</sup>A partir d'una solució inicial es selecciona una nova, aleatòriament, pròxima a la inicial. Si es millor s'hi queda, i sinó, segons una certa probabilitat, torna a l'anterior o es queda en la nova. Això es repeteix fins a la condició d'acabament[60].



### 1.6.6 Aprenentatge per reforç (*RL*)

En la robòtica, l'aprenentatge per reforç proveeix d'unes eines molt útils per tal de crear comportaments sofisticats i amb gran dificultat de disseny. Permet a un robot desenvolupar el seu propi comportament a base de prova i error. En aquest cas, el dissenyador, en lloc de donar unes dades per explícitament crear la solució al problema, tan sols proveeix una realimentació amb una funció objectiu de valors escalars que mesura la bondat de l'acció anterior. Per tant, un agent explora les possibles estratègies i després rep una recompensa per l'acció feta, intentant sempre maximitzar la recompensa acumulada durant el seu temps de vida [23]. Però a diferència de l'aprenentatge supervisat, no es “*diu*” quina acció hauria estat la millor a llarg plaç, a més de no haver d'explorar l'entorn, una altra diferència, en *RL* el fet de les accions ser en temps real és molt influent, ja que és concurrent amb l'aprenentatge [20].

Aquest agent i el seu entorn poden ser modelat com un estat<sup>6</sup>  $s \in S$  i una acció<sup>7</sup>  $a \in A$ . Una recompensa es donada a l'agent, per cadascuna de les accions que desenvolupa, en funció de l'estat i les observacions. L'objectiu de *RL* és crear una política<sup>8</sup>  $\pi$  que maximitza la recompensa acumulada escollint unes accions  $a$  en determinats estats  $s$ .

La idea clàssica d'aprenentatge per reforç es pren des del punt de vista que l'agent consistia en un procés de decisions de Markov (*Markov Decision Process* o *MDP*)<sup>9</sup> on la propietat de Markov estableix que el següent estat  $s'$  i la recompensa estan definits tan sols per l'acció  $a$  i l'estat  $s$  [57].

L'acumulació de recompensa és el que es maximitza o minimitza segons l'algorisme utilitzat, aquí s'exemplifica maximitzant. Per tant segons el mètode d'atorgar la recompensa es defineix el comportament òptim [23]. Existeixen diversos models, aquí se n'exposen tres:

**Horitzó finit** Aplicat en models on es sap en quants passos es resol el problema, maximitza la recompensa per  $H$  passos.

$$J = E \left\{ \sum_{h=0}^H R_h \right\}. \quad (6.4)$$

**Model de descompte** Un factor de descompte ( $\gamma \in [0, 1)$ ) a la recompensa futura.

<sup>6</sup>Un estat  $s$  conté la informació necessària per descriure la situació actual i futures.

<sup>7</sup>Un estat del sistema es controlat o carregat per una acció  $a$ .

<sup>8</sup>Per política s'entén com en [10] “*Manera de conduir un afer*.”

<sup>9</sup>Conjunt d'estats  $S$ , accions  $A$ , recompenses  $R$  i probabilitats de transició  $T$ , aquest últim defineix la dinàmica del sistema per predir l'efecte de l'acció en un estat donat.

Aquest és introduït manualment i determina en quina proporció afecta el futur<sup>10</sup>.

$$J = E \left\{ \sum_{h=0}^{\infty} \gamma^h R_h \right\}. \quad (6.5)$$

**Recompensa mitja** Finalment en aquest es té en compte la mitja total de les recompenses. El problema d'aquesta és que no es pot diferenciar si s'esta afavorint l'inici o el final del temps de vida.

$$J = \lim_{H \rightarrow \infty} E \left\{ \frac{1}{H} \sum_{h=0}^H R_h \right\}. \quad (6.6)$$

A l'hora d'estimar la política  $\pi$  òptima, existeixen una gran diversitat de mètodes que es podrien desglossar en dos grans grups segons si requereixen del model probabilístic de transició  $T(s', a, s)$ .

- Els mètodes que requereixen de model són anomenats *model-based*.
- Per altra banda, dels que no requereixen d'un model els més utilitzats són el *Monte Carlo*, Mètodes de diferencia temporal, *SARSA*, *R-learning* i *Q-learning*, sent aquest últim el més extès, gràcies a la seva senzillesa.

El comportament après és totalment dependent de la funció de recompensa que s'ha utilitzat. En la practica, és molt difícil crear la funció per a l'aprenentatge per reforç d'un robot. En molts cops, convé utilitzar recompenses contínues per tal de guiar l'aprenentatge, en lloc d'una recompensa binària segons si s'ha complert o no la tasca [24]. Molts cops el comportament no és l'esperat, tot i que, per la nostra forma de pensar semblés que la solució és òbvia. Per això, en alguns casos, s'utilitza l'aprenentatge per reforç invers, que aconsegueix extreure la funció recompensa gràcies a un seguit de demostracions, pot ser no sigui la verdadera recompensa, però provoca l'actuació de la forma desitjada [23].

Alguns dels molts problemes que comporta el fet d'aplicar-se en robots es descriuen a [23]. A banda d'haver de decidir la forma de treballar: com d'acurat es vol el control del robot, si discret o per aproximació de funcions, a quina freqüència actuar, etc. Un ha de tenir en compte que l'augment de la dimensionalitat provoca un creixement exponencial dels càlculs per cobrir l'espai d'estats i accions, és per això que en molts cops es treballa l'aprenentatge de forma jeràrquica<sup>11</sup> o amb tasques progressives<sup>12</sup>. Per altra banda, dur

<sup>10</sup>Com més pròxim a 0 la recompensa a llarg plaç és menys significant, ara bé, també s'ha de tenir en compte que la *policy* òptima pot ser inestable si el factor de descompte és massa baix [23].

<sup>11</sup>S'assumeix que una part és fixa, mentre les altres s'aprenen, per tenir un solució inicial per després fer l'aprenentatge global.

<sup>12</sup>Alguns cops és més senzill aprendre una tasca complicada si es fan anteriorment algunes de no tant

a terme experiments en el món físic és car, la comunicació i reacció dels motors del robot porten sempre un cert retard, pot ser complicat el recrear les condicions de l'entorn necessàries per l'aprenentatge i s'ha de tenir molta cura perquè l'exploració d'aquest entorn sigui segur, ja que pot crear tot tipus de riscos. Per a molts d'aquests problemes, la solució podria ser l'ús de models simulats, però s'ha de tenir en compte que aquests no són perfectes i tan sols un petit error pot acumular i donar un comportament diferent.

### 1.6.7 DMP (*Dynamic Movement Primitive*)

Les *DMPs* representen un moviment a partir d'un conjunt d'equacions diferencials, on la dinàmica del propi sistema corregeix les pertorbacions que puguin aparèixer, és per això que són considerats sistemes robusts davant pertorbacions. A més, és molt fàcil modificar l'objectiu (o *goal*) del moviment al estar presentat com equacions, ja que tan sols és modificar el paràmetre  $g$  en l'equació. Aquesta robustesa i adaptabilitat que dona aquest tipus d'entorn de treball és molt favorable per millorar altres sistemes d'aprenentatge com és l'aprenentatge per demostració (*learning from demonstration* o *LfD*) on a partir de certs exemples s'aprén el comportament.

El *LfD* és podria estructurar en tres grans grups, segons la forma d'adquirir les dades de la demostració:

**Imitació** L'exemple és produeix sobre una plataforma que no és el robot, per tant, la informació extreta requereix ser modificada i interpretada per adequar-se a les articulacions del robot. Dos possibles mètodes són amb sensors a sobre el professor o a través de l'observació externa amb els sensors del robot.

**Demostració** L'execució és produeix sobre el mateix robot, per tant, no s'han de transformar les dades per tal d'interpretar com és el moviment sobre els motors del propi robot. En aquest cas, un dels mètodes utilitzats és la teleoperació del robot per part del professor, mentre l'autòmat registre el moviment amb els sensors propis.

**Trajectòria programada** Per últim, la demostració pot ser donada per un seguit de coordenades d'una trajectòria preestablerta en el propi codi. Aquest cas, només es possible d'efectuar si el robot "sap" en tot moment la posició de les seves articulacions, i per tant, és pot complir perfectament el recorregut.

---

complicades.

## Transformation system

El sistema dinàmic és pot interpretar com un PD<sup>13</sup>, amb els paràmetres  $K$ , pel coeficient proporcional, i  $D$ , pel derivatiu; o com si fos un sistema mecànic de molla lineal amb una força externa viscosa, en aquest cas, sent el coeficient de fregament i de la fricció viscosa respectivament. Per últim, la  $x$  i  $v$  són la posició i la velocitat, la constant  $\tau$  és el període del moviment i  $g$  és el paràmtre d'atracció del sistema.

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \quad (6.7)$$

$$\tau \dot{x} = v \quad (6.8)$$

Si s'estudia el sistema dinàmic unidimensional de les equacions (6.7) (6.8), que correspondrien al sistema de transformació (*transformation system*), és pot comprovar que aquest és estable, tendint sempre a la posició  $g$ , per qualsevol valor de  $f(s)$ . Aquesta és una funció no lineal que no depèn del temps, sinó de la variable de fase  $s \in [0, 1]$  que representa la durada del moviment en tant per un. Aquesta variable està definida per  $\tau$  i per  $\alpha$ <sup>14</sup> com es veu en l'equació diferencial (6.10), conegut com a sistema canònic (*canonical system*). A més, aquesta funció  $f(s)$  pot aprendre per tal de dur a terme moviments complexes de forma arbitrària, ja que els pesos  $w_i$  es poden ajustar.

$$f(s) = \frac{\sum_i w_i \psi_i(s) s}{\sum_i \psi_i(s)} \quad (6.9)$$

$$\tau \dot{s} = -\alpha s \quad (6.10)$$

Les  $\psi_i(s)$  són funcions gaussianes expressades com  $\psi_i(s) = \exp(-h_i(s - c_i)^2)$  on les  $h_i$  defineixen l'amplada i les  $c_i$  el centre de la gaussiana  $i$ . La peculiaritat de la funció  $f(s)$  és el fet de ser la suma ponderada de les gaussianes, cadascuna amb el seu pes, i per tant, pot crear la corba que és vulgui, com es veu exemplificat en la figura 6.8a. Això permet que, aquesta corba no lineal, sigui sumada amb la trajectòria, definida pels paràmetres  $K$  i  $D$ , com en l'exemple de la figura 6.8b, per així, poder-se adaptar a noves situacions (com evadir obstacles, canvi d'objectiu, etc.).

Realment, en les *DMPs*, el que determina la trajectòria duta a terme és la part no lineal, i aquesta es veu controlada per els pesos  $w_i$ . Per aprendre aquests existeixen dife-

<sup>13</sup>Controlador proporcional i derivatiu.

<sup>14</sup>La  $\alpha$  és una constant pre-definida

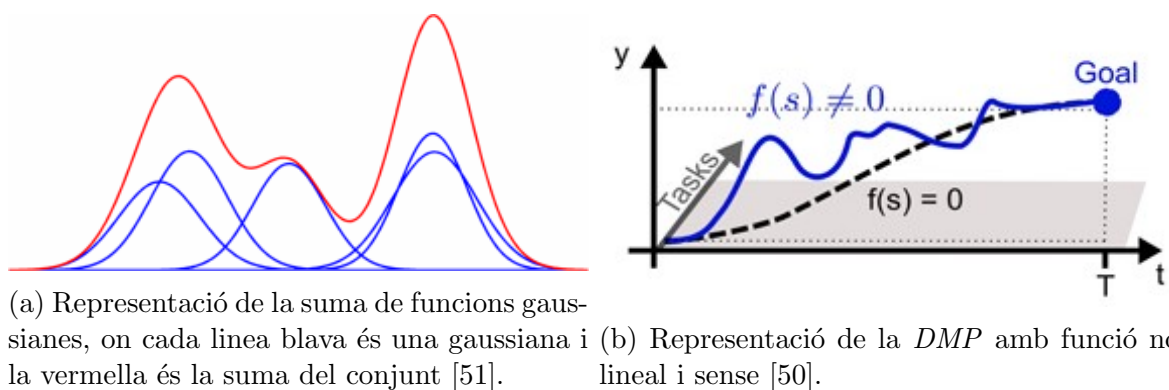


Figura 6.8: Gràfics explicatius de la funció no lineal  $f(s)$

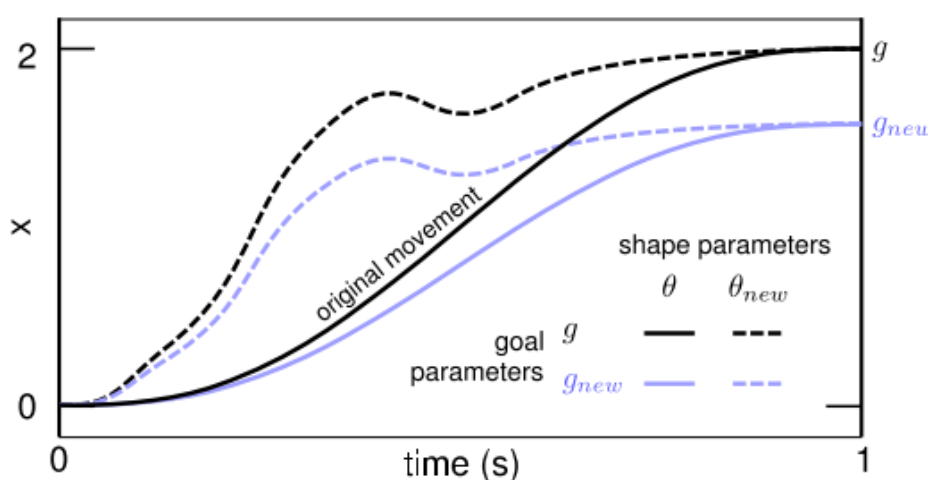


Figura 6.9: Efecte dels pesos  $w_i$  (representats per  $\theta$ ) i de l'objectiu (*goal*) en la trajectòria [55]. Els pesos varien l'estil de la trajectòria, mentre el *goal*, l'escurça o allarga.

rents mètodes: (1) aprenentatge per demostració, on es fa una aproximació, per mínims quadrats, amb gaussianes d'aquesta trajectòria; (2) aprenentatge per reforç, en aquest cas, un dels algorismes més utilitzats per les DMPs és el  $PI^2$ , que s'explica en el següent punt d'aquesta secció.

## Modificació de l'algorisme

Fins ara s'ha explicat l'algorisme original de les DMPs, ara bé, aquest porta inherents una sèrie d'inconvenients [39]:

- Si la posició inicial  $x_0$  i la posició  $g$  són la mateixa, llavors la funció  $f(s)$  no es capaç de desplaçar el sistema de l'estat inicial.
- Si es dona el cas que  $g - x_0$  és molt pròxim a zero, probablement  $f(s)$  sigui un

numero elevat, per tant, si varia el valor de  $g$  pot provocar acceleracions molt grans que sobrepassi els límits del robot.

- Finalment, s'ha de tenir en compte el fet que si el signe de  $g_{new} - x_0$  és canviat respecte  $g_{original} - x_0$  l'efecte de la funció no lineal és reflectit.

Es per això, que en l'article [39] es proposa una modificació de l'algorisme original. L'única equació que es veu retocada és (6.7) que és substituïda per

$$\tau \dot{v} = K(g-x) - Dv - K(g-x_0)s + Kf(s) \quad (6.11)$$

Els trets importants són que  $g - x_0$  no multiplica a  $f(s)$  i el terme  $K(g - x_0)s$  és necessari per tal que en l'inici del moviment no es produeixin salts, la millora, respecte l'algorisme original, es veu reflectida a la Figura 6.10 de sobre.

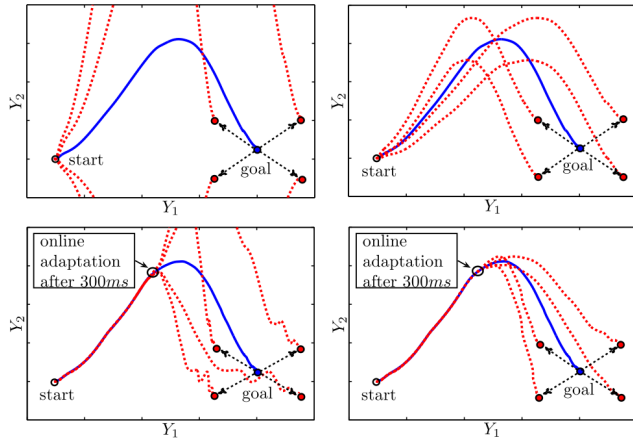


Figura 6.10: Comparació de l'algorisme de *DMP* original (esquerra) i el modificat per [39] (dreta). S'utilitzen els mateixos moviment original i *goals*, a dalt els goals són modificats a l'inici, mentre a la part de baix es modifiquen als 300 ms d'haver començat.

### 1.6.8 Path Integral Policy Improvement (PI<sup>2</sup>)

L'algorisme PI<sup>2</sup> [55] té com a principal objectiu polir el pesos  $w_i$  (al llarg d'aquest apartat s'hi refereix com paràmetres  $\theta_t$ ), per tal que es minimitzi la funció de cost (6.12) de la trajectòria  $\tau_i$ . Ara bé, una bona solució inicial, per aconseguir una convergència més ràpida, és l'extreta a partir del LfD, comentat en el punt anterior.

$$J(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2} \theta_t^T R \theta_t) dt \quad (6.12)$$

La funció  $J(\tau_i)$  és creada per l'usuari, segons la tasca que es vulgui desenvolupar, sent  $\phi_{t_N}$  el cost final,  $r_t$  el cost immediat i  $\frac{1}{2} \theta_t^T R \theta_t$  cost de control immediat<sup>15</sup>. Aquests dos últims, determinen principalment els valors que prenen els diferents  $\theta_t$  durant la

<sup>15</sup>Regula el fet d'augmentar o disminuir els paràmetres  $\theta_t$ , envers el benefici de fer-ho [14]

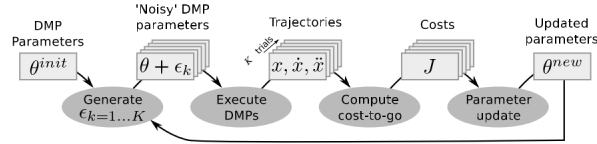


Figura 6.11: Algorisme genèric dels mètodes de millora de política [55]

trajectòria, mentre el cost final  $\phi_{t_N}$  és el que decideix la bondat del resultat. Degut a la pròpia naturalesa d'aquests, el cost final ha de ser el més influent, ja que normalment el que interessa més és arribar a l'objectiu. Per això, quan la tasca que interessa és que en un determinat instant el robot passi per una coordenada en concret<sup>16</sup>, els costos es presentarien d'una forma similar a (6.13). L'altre plantejament seria com un problema de RL, per exemple<sup>17</sup>, si es vol acabar en un punt (*goal*), amb poca velocitat i durant el recorregut amb l'acceleració minimitzada seria de l'estil de (6.14).

$$r_{300ms} = 100000000(G - y_{t_{300ms}})^2 \quad \phi_{t_N} = 0 \quad (6.13)$$

$$r_t = 0.5\ddot{y}_t^2 + \frac{1}{2}10000(\theta_t^T \theta_t) \quad \phi_{t_N} = 10000(y_{t_N}^2 + (g - y_t)^2) \quad (6.14)$$

Els mètodes de millora de política, com és el PI<sup>2</sup>, consisteixen en un procés iteratiu d'exploració i actualització dels paràmetres. En l'exploració es proven  $K$  *DMPs* diferenciades per prendre els  $\theta^{init}$  inicials més un cert soroll  $\epsilon_{t,k}^\theta$ , d'aquests se'n calcula les respectives funcions de cost i a partir d'aquí s'actualitzen els valors dels paràmetres per donar  $\theta^{new}$  que són els actuals.

Per últim, en el cas concret del PI<sup>2</sup>, l'algorisme segueix la següent estructura, segons [55] on està més detallat:

1. Determina el cost de cadascuna dels **trials**, per tant, du a terme cada *DMP* amb el respectiu paràmetre  $\theta^{init} + \epsilon_{t,k}^\theta$ .
2. Calcula la probabilitat de cada un. La idea és que a menor cost, major probabilitat.
3. Proceix a calcular el  $\delta\theta$ , primer és traient la mitjana respecte els **trials**, on es té en compte la probabilitat d'aquests, i després la mitjana respecte el temps.
4. Finalment dona  $\theta^{new} = \theta^{init} + \delta\theta$ .

<sup>16</sup>Extret de [52] apartat 5.2 *Learning Optimal Performance of a 1 DOF Via-Point Task*.

<sup>17</sup>Extret de [52] apartat 5.1 *Learning Optimal Performance of a 1 DOF Reaching Task*.

### 1.6.9 Algorismes avançats

**CPG** *Central Pattern Generators*, la idea és crear una arquitectura capaç de generar coordinació entre diferents elements, independentment de la tasca a realitzar i de la plataforma robòtica utilitzada. Una forma de veure-ho és des del punt de vista dels autors de [58]:

*"...we see the robot's mind as a group of different modules each one in charge of its own device (sensor or actuator) that interacts with the rest of modules..."*

Aquesta cita podria ser traduïda com que cada dispositiu encarregant-se d'ell mateix, però amb la interacció amb els altres, tots junt arriben a crear la ment del robot.

Per poder dur a terme aquesta arquitectura es requereixen de dos tipus d'algorismes: (1) algorismes neuro-evolutius, per poder cooperar entre mòduls i controlar els elements associats; i (2) algorismes co-evolutius, per instruir i arribar a un objectiu comú entre tots.

**CBR** *Case Based Reasoning*, aquest algorisme podria ser considerat de la família de l'aprenentatge supervisat. Consisteix l'aproximació de l'acció correcte a través de dos tipus de dades: (1) dades d'entrenament, del mateix estil que les del supervisat; i (2) extrems a partir de la pròpia experiència. El cicle de funcionament del CBR seria el següent: (i) prendre el cas o els casos més semblants a la situació actual, dels que estan emmagatzemats; (ii) adaptar el cas pres a la situació; (iii) avaluar com de satisfactori ha estat la solució adoptada; (iv) aprendre d'aquest nou cas.

**GA** *Genetic Algorithm*, és un mètode estocàstic de cerca que pren la idea de l'evolució biològica natural. Aquest pren uns antecedents aleatoris, d'aquests en treu solucions les quals s'hi provoca una mutació, per últim, les solucions alterades es converteixen en els antecedents. Aquest procés es repeteix fins arribar a la solució que s'adapta suficient a la funció objectiu o al limit de generacions.



## Capítol 2

# Estudis preliminars

### 2.1 Comunicació mitjançant *ROS*

L'entorn de treball *ROS* és una eina que està a l'ordre del dia en robòtica. Fins aquest moment, no existia la possibilitat de comunicació del robot AIBO a través de *ROS*, tot i que han existit intents, però no serveixen pel nostre cas<sup>1</sup>. Per això, tot seguint una iniciativa del Departament d'Automàtica de la Universitat Politècnica de Catalunya, el doctor Ricardo Téllez va fer-se-la seva i va començar a crear un primer **package**. Des del Departament, i a través de diversos TFGs, s'ha promogut la continuació d'aquest **package** per permetre que l'AIBO pugui treballar amb *ROS*. Per això, dos Projecte de Final de Carrera i aquest mateix Treball Final de Grau, han millorat de forma conjunta aquesta eina.

Gracies a aquests ara permet la transferència de video i so, el control de les articulacions del robot, s'ha optimitzat la freqüència de treball, etc. Actualment la connexió amb AIBO comporta la creació de sis **topics** on aquest publica i dos al que està com a subscriptor, la llista és:

- Publicador en els **topics**:

**/aibo/infrared** on dona la informació de cada sensor d'infrarojos (**distanceChest**, **distanceNear**, **distanceFar**), la distància que ha calculat fins un obstacle en cadascun.

**/aibo/image** en aquest **topic** publica les imatges de la càmera frontal de l'AIBO,

---

<sup>1</sup>Per una banda, el cas del **roseus package** [37] que utilitza l'entorn de treball **EusLisp**, no ha tingut gaire èxit. Per l'altra banda, s'ha enfocat des de perspectives diferents, a les establertes en aquest treball, com en el cas de [48].

en format `sensor_msgs::Image`.

`/aibo/joints` en aquest `topic` es publiquen les posicions de cadascuna de les articulacions en unitats de graus.

`/aibo/accel` les acceleracions de l'acceleròmetre que porta incorporat l'AIBO en unitats de  $m/s^2$ .

`/aibo/paws` els sensors de les potes són binàris, per tant els valors d'aquest `topic` varien entre 1 i 0 segons si estan presionats o no.

`/aibo/touch` en aquest cas els sensors són de pressió, excepte el quint valor que també és binari.

- Subscriptor dels `topics`:

`/aibo/sound` en aquest `topic` s'hi publiquen els sons que desitgen ser reproduïts per l'AIBO.

`/aibo/subJoints` quan es desitja que l'AIBO prengui una determinada posició, definida per l'estat de les articulacions, s'ha de publicar en aquest `topic` un missatge de tipus `aibo_server::Joints`<sup>2</sup> amb les unitats de graus restringit dins el rang determinat en la taula 6.1.

### 2.1.1 Entre l'AIBO i *ROS*

El funcionament actual de la comunicació entre AIBO i *ROS* es basa en Urbi, ja que la informació que es publica en els `topics` de *ROS*, en realitat, són les dades extretes d'Urbi 1.5<sup>3</sup>. De la mateixa manera, les accions de control que són publicades a *ROS* s'interpreten i es transformen en comandes d'Urbi. Aquests conceptes s'expliquen a continuació en detall i es veuen il·lustrats en la Figura 1.1. Aquests dos tipus de tasques es desenvolupen en paral·lel per dos clients d'Urbi separats: (1) client publicador, és aquell que introdueix les dades de l'AIBO en els `topics`; (2) client subscriptor, on la seva funció és recollir el que s'ha publicat en el `topic` de comandes.

**Publicació a *ROS*** d'informació de l'AIBO a traves d'Urbi

1. Per una banda, Urbi demana la informació a l'AIBO, en forma de `callbacks`.

<sup>2</sup>Aquest tipus de missatge està inclòs en el `package aibo_server`.

<sup>3</sup>S'ha de remarcar el fet que és l'Urbi 1.5 perquè per a versions posteriors ja existeix la comunicació entre Urbi i *ROS*.

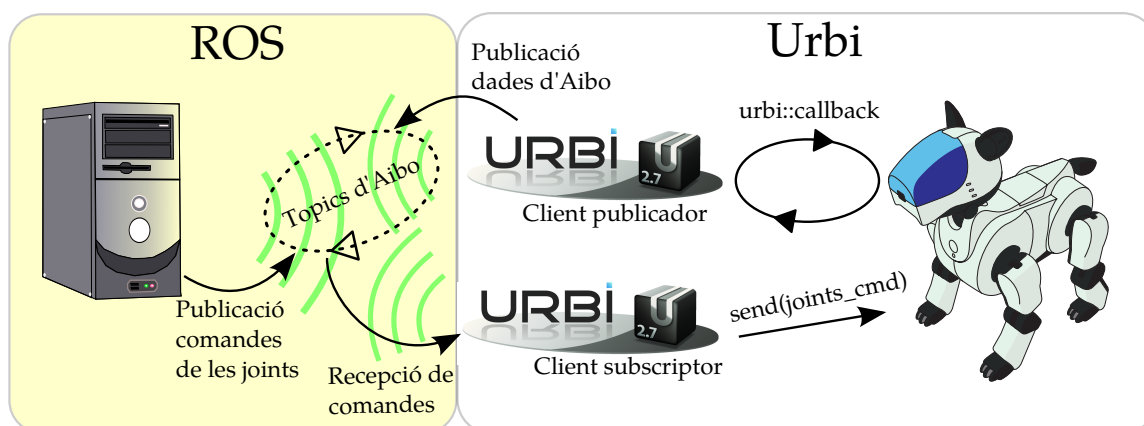


Figura 1.1: Comunicació entre l'AIBO, Urbi i ROS

2. Per l'altra banda, Urbi publica continuament en els **topics**, on hi ha subscriptors, el missatge respectiu segons el **topic**.
3. En el moment en que la informació de l'AIBO és rebuda per Urbi, es modifica el missatge que aquest publica a ROS.

### Control de les articulacions a partir d'un topic

1. L'algorisme de control crea un **topic** anomenat `aibo_server/aibo/subJoints`, cada cop que s'hagi de modificar la posició de les articulacions, hi publica un **message**<sup>4</sup>.
2. En aparèixer un nou **message** al **topic**, Urbi pren el **message** i el transforma per tal d'enviar-lo com a comanda per les articulacions a l'AIBO.

### 2.1.2 Entre l'algorisme de control i ROS

En parlar d'algorisme de control es refereix al codi que processa els càlculs que es requereixen per dur a terme tot el procés d'estabilització. Aquest codi és executat en un ordinador o servidor, on està instal·lat tot el *software* requerit<sup>5</sup>.

L'algorisme de control només depèn de ROS per recollir la informació del conjunt acceleròmetre-giroscopi i per executar el moviment de les articulacions de l'AIBO. Per tant, el **node** d'execució és tant subscriptor del **topic** del sensor, com, alhora, és publicador del **topic** de les comandaments per les articulacions.

<sup>4</sup>El **message** conté un conjunt de **floats** que representen cadascuna de les articulacions, amb el seu nom seguint la Taula 6.1, canviant la paraula "leg" per "joint".

<sup>5</sup>El procés d'instal·lació està explicat en els Annexos.

### 2.1.3 Entre l'Arduino i *ROS*

El conjunt està connectat a l'Arduino, per tant, ha de ser aquest el que creï el `topic` on es publica la informació que ha de transmetre per *ROS* a l'algorisme de control. A tal efecte existeix un `package` que permet aconseguir-ho, `rosserial`. A banda d'haver-se d'instal·lar i executar quan és el cas, també es requereix que en el codi es carreguin la llibreria `ros.h` i les corresponents al tipus de `message` al que s'hagi de subscriure o de publicar. Aquestes llibreries, juntament amb les variables que es requereixen com el `node`, el publicador i el `message`, ocupen una gran quantitat de memòria del microcontrolador i pot arribar a donar algun que altre problema, com s'explica més endavant en l'apartat 3.2.2.

## 2.2 Estabilitat

El concepte d'estabilitat és molt genèric, s'ha definit de múltiples maneres al llarg del temps, sense mai tenir una única interpretació. Dins el concepte d'estabilitat es pot diferenciar entre dues:

**Estàtica** és aquella que assumeix que la projecció vertical del *CdG* està, en tot moment, dins el polígon d'estabilitat<sup>1</sup> amb un marge d'estabilitat<sup>2</sup> adequat [17].

**Dinàmica**, en aquest cas, es tenen en compte els efectes de paràmetres dinàmics com la velocitat i l'acceleració de diferents parts del robot [62].

El fet d'estudiar l'estabilitat dinàmica es sortiria de l'abast d'aquest treball, per la seva gran complexitat, tot i existir conceptes com el *ZMP*, explicat en detall en [61], que permeten aquests tipus d'estudis, fins i tot per robots bípedes. Però en aquest cas, tan sols es treballa amb l'estabilitat estàtica. Per tant, s'estableix que l'objectiu de l'algorisme de control és mantenir el *CdG* del robot a dintre del polígon d'estabilitat en tot moment. Aquesta tasca es podria aconseguir realitzar de diferents formes, però la majoria d'aquestes probablement queden fora de l'abast d'aquest treball. Per això, l'idea és fer-ne una aproximació, s'assumeix que la plataforma no arriba, en cap cas, a certes pendent molt elevades<sup>3</sup>.

<sup>1</sup>El polígon d'estabilitat està definit pels punts de suport del robot.

<sup>2</sup>El marge d'estabilitat assegura que per qualsevol velocitat que el robot pot arribar, no caurà degut al seu propi moment [17].

<sup>3</sup>Aquesta pendent màxima s'estableix en els objectius del *TFG*, Secció 1.1

## 2.3 Algorismes de resposta davant pertorbacions

En la robòtica, com en qualsevol àmbit, per un mateix problema poden ser utilitzades infinitat de solucions. Ara bé, el tret característic de l'enginyeria és que d'entre la multitud de possibilitats, s'esculli la més òptima segons les condicions del moment. Abans d'optar per una opció, s'ha de tenir una idea clara del que aporta cadascuna i observar com s'adapta a la problemàtica actual.

En l'estat de l'art, s'han esmentat algunes de les possibilitats per dur a terme els objectius fixats inicialment. Aquestes opcions han estat explicades anteriorment, amb una breu descripció i trets característics que podrien ser d'interès pel nostre cas. A continuació, s'exposa com cadascuna podria adaptar-se al problema, mencionant els seus avantatges i inconvenients.

### 2.3.1 Model del robot

Una de les possibilitats podria ser la creació d'un model del l'AIBO. D'aquesta forma davant una pertorbació, com és el canvi d'inclinació de la plataforma, es podrien saber les accions de control que portarien el robot a l'estat desitjat.

A banda de beneficiar-se de la cinemàtica inversa<sup>1</sup>, un model permetria aplicar els algorismes, de forma virtual, en un simulador. L'algorisme simulat donaria una solució, que podria servir com a punt de partida per aplicar-ho al model físic, així la probabilitat de provocar algun risc seria molt més baixa.

### 2.3.2 Aprenentatge supervisat

Per altra banda, per no continuar amb la forma típica de controlar els robots com és el fer un model, es podria utilitzar l'aprenentatge supervisat. Per tant, a partir d'un conjunt de mostres d'exemple que utilitzi com a variables explicatives l'estat del robot i com a resposta la posició final de les articulacions. Com estat del robot s'hauria d'interpretar com el conjunt de posicions de les articulacions i inclinacions, en l'eix x i l'eix y, actuals. S'han d'utilitzar tant posicions com inclinacions, ja que sinó un mateix estat tindria més d'una solució possible.

Es cert que si s'aconsegueix determinar de forma correcte l'espai de solucions, aquest model podria ser molt útil. Ara bé, hi ha cert inconvenients a l'hora de crear les mostres

---

<sup>1</sup>Tècnica permet determinar el moviment d'unes articulacions per portar el cos a una posició concreta

d'exemple segons [2]:

- Pot haver-hi impresscions en la gravació d'atributs d'entrada. En aquest cas és l'error que tenen els sensors de les articulacions i l'aparell que mesura l'angle d'inclinació.
- Possibles errors a l'hora d'etiquetar la resposta, ja que és difícil, a banda d'infinites solucions<sup>2</sup>, encertar a mà quines han de ser les posicions exactes perquè l'AIBO es trobi horitzontal. A més, no hi ha una mesura exacta de com d'estable es troba el robot, l'únic que es té es la inclinació, però podria estar desplaçat.
- Per últim, poden existir factors que no s'han tingut en compte, però que influeixin la resposta, com podria ser el desplaçament lateral o longitudinal, o l'acceleració.

Per la pròpia motivació que es té en el treball de voler aprendre nous coneixements, i que aquest mètode en el seu origen és fer un estudi estadístic, s'ha preferit no utilitzar-lo, tot i poder ser una eina totalment vàlida.

### 2.3.3 Aprenentatge per reforç

D'entre les possibilitats que s'han plantejat fins aquí en aquest apartat, aquesta és la que més ha atret. Això és degut a que és una eina molt potent, però amb una lògica interna mitjanament simple. S'hauria de plantejar el mètode que més s'adapti al cas, i llavors crear la funció a minimitzar o maximitzar per tal de dur a terme la tasca d'estabilitat.

En aquest treball, no es té un model del robot prou acurat, per tant, es requereix d'un algorisme apta per ser utilitzat sense model. D'entre els possibles, s'escolliria el **Q-learning**, ja que és del que es disposa de més informació i, a més, l'algorisme no és molt difícil d'implementar. El **Q-learning** es basa en augmentar o disminuir la probabilitat d'una acció en un estat concret, segons el **reward** que s'ha concedit quan s'ha fet aquesta acció en aquell estat en un instant del passat<sup>3</sup>.

En referència a l'acumulació de recompensa  $J$ , que és la funció a maximitzar o minimitzar, s'hauria d'escollir inicialment quin model utilitzar d'entre: (1) horitzó finit, (2) model de descompte o (3) recompensa mitja. Al ser l'objectiu la bondat de la posició final i que és duguí a terme en un temps determinat, el model que s'hi adapta millor és

<sup>2</sup>Existeixen infinites solucions ja que en cada pota hi ha dues articulacions que permeten desplaçar o inclinar el robot de forma longitudinal.

<sup>3</sup>Per tant, en tornar al mateix estat hi ha més probabilitats de fer l'acció correcta, o com a mínim, menys probabilitat d'equivocar-se de nou.

el d'**Horitzó finit**:

$$J = E \left\{ \sum_{h=0}^H R_h \right\}. \quad (3.1)$$

Els algorismes d'aprenentatge per reforç es basen en gran part en l'exploració dels estats, ara bé, en un sistema com és l'AIBO existeixen massa possibles estats per poder explorar-los de forma física, a banda que són continus, i per tant, s'haurien de discretitzar per utilitzar algorismes com **Q-learning**. Per això, si s'hagués de fer s'utilitzaria l'aprenentatge de forma jeràrquica, per tal de primer només aplicar-ho a unes poques articulacions, mentre les altres estan fixes i aquestes s'afegirien de forma progressiva quan les primeres ja hagin après.

Ara bé, tot i poder-se dur a terme seguint aquest procediment, és creu més convenient utilitzar algun altre mètode que convergeixi de forma més ràpida, i si escau, fer ús del RL per millorar el comportament del que s'hagi aconseguit.

### 2.3.4 *DMP*

L'algorisme de *DMP* és el que s'adapta millor a la situació, ja que està dissenyat per tractar sistemes dinàmics, com és el cas. A més, té una gran robustesa davant pertorbacions, que és el que justament es necessita, considerant que el moviment de la plataforma és una pertorbació. Per altra banda, s'adapta perfectament a noves situacions, com pot ser un inici diferent al original i/o acabar en una posició que difereixi de la posició final primera.

El robot consta de diferents articulacions, cadascuna afecta d'una o altre manera el moviment del conjunt del robot. Per implementar l'algorisme, s'ha de crear una *DMP* per cadascuna de les articulacions, per tant el procés d'aprenentatge s'ha de repetir tants cops com nombre junctures. En la majoria de les explicacions següents és refereix en l'aprenentatge d'una individualment, per les altres seria el mateix procés.

#### Moviment original

El primer pas que s'ha de fer, per dur a terme la *DMP*, és tenir un moviment original. Aquest és podria extreure, com s'ha explicat en l'apartat 1.6.7, a partir de: (1) imitació, (2) demostració amb el propi robot, (3) trajectòria programada. D'entre les tres, s'ha escollit la tercera, ja que les articulacions de l'AIBO són *PIDs* i, per tant, la posició que

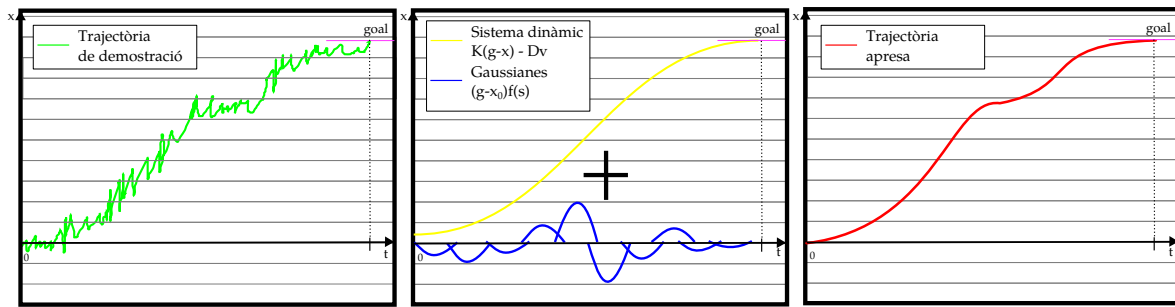


Figura 3.1: Donada una trajectòria de demostració (esquerra), aquesta és aproximada a partir de la suma del sistema dinàmic, definit per  $K$  i  $D$  preestablertes, i d'unes gaussians (centre), que són les que s'han aproximat per mínims quadrats. La suma resulta en la trajectòria apresada (dreta).

se li programi és segur que es durà a terme. A més, la imitació no tindria sentit, si no hi ha un altre robot a imitar, i la demostració no és una bona opció, perquè a banda de ser impossible donar el moviment a totes les articulacions alhora de forma manual, tampoc seria precís.

Havent escollit el mètode per donar el moviment original, s'ha de saber com ha de ser aquest. Un tipus de trajectòria molt utilitzat és la de mínim *jerk*<sup>4</sup>. Aquesta és una trajectòria suavitzada calculada a partir d'una posició inicial, final i el temps requerit per arribar-hi, minimitzant el *jerk*. Es deixen els detalls d'aquest tipus de trajectòria per [54].

### Pesos de la funció $f(s)$

Un cop es té la trajectòria, a partir d'aquesta es poden extreure els pesos  $w_i$  aproximant mínims quadrats amb gaussians i, per tant, la funció no lineal  $f(s)$  queda definida amb una primera solució, figura 3.1. Amb tan sols aquesta primera solució és factible el dur a terme les *DMPs* de forma correcta. Ara bé, en el treball és desitja que s'arribi a la posició de màxima estabilitat de la forma més precisa. Per això, s'han de poder modificar els pesos  $w_i$  i el *goals* de les articulacions, de tal manera que la posició final sigui l'adequada. Una bona opció és l'algorisme  $PI^2$ , que et permet millorar tan els pesos, com els *goals* dependent de paràmetres externs a la dinàmica del sistema, com podria ser el desplaçament del *CdG* o la inclinació.

<sup>4</sup>*Jerk* és la derivada de l'acceleració



### 2.3.5 Elecció final

En conclusió, el mètode utilitzat és la *DMP* conjuntament amb l'algorisme  $PI^2$ . Per una banda, la *DMP* dona una solució inicial bastant encertada<sup>5</sup>. Per l'altra banda, el  $PI^2$  millora tant els pesos com els *goals* de les articulacions, depenent dels paràmetres externs com són el desplaçament del *CdG* o la inclinació del robot.

## 2.4 Codis amb *DMPs* implementades

La creació de l'algorisme de *DMPs* és un fet que no troba dins l'abast del treball, degut a la gran complexitat d'aquest. Està basat en equacions diferencials, canvis de variables, ús de Gaussians o Fourier (segons la implementació), aproximacions per mínims quadrats, etc. Per una banda, estaria la dificultat de crear el codi, però més complicada és el treball que porta el depurar-lo. Per això, s'ha optat per cercar codis que portin implementades les *DMPs*, d'entre els que s'han trobat només s'han plantejat els dos que es troben a continuació perquè són uns dels que utilitzen la *DMP* modificada que s'ha plantejat en l'apartat 1.6.7 de l'estat de l'art.

### 2.4.1 *DMPs* de Scott Niekum

El package creat per Scott Niekum està basat en l'article [39]. Inicialment, s'ha optat per aquest codi perquè és senzill i útil, figura 4.1. S'ha estudiat el seu funcionament, on a diferència de l'algorisme explicat en l'apartat 1.6.7, aquest l'aproximació es fa amb *Fourier*, en lloc d'utilitzar *gaussianes*. D'aquest codi, se'n poden diferenciar dues grans parts: (1) l'aprenentatge dels pesos a partir de la trajectòria donada, (2) el càlcul de les accions de control del robot, com són posicions i velocitats suposades les articulacions en cada instant.

#### Aprenentatge dels pesos

A l'hora de fer l'aprenentatge es prenen un seguit de  $m$  mostres de la trajectòria, d'on de cada trajectòria  $j$  s'extreu la posició  $x_j$  i velocitat  $v_j$  en l'instant  $t_j$ . A partir d'aquestes variables es calcula la  $f_{target}(s_j)$  en (4.2) que és el valor de  $f(s_j)$  per dur a terme la mateixa trajectòria que la d'exemple. El valor de  $s_j$ , calculat en (4.1), sorgeix

<sup>5</sup>La bondat d'aquesta solució depèn de la qualitat del *goal* per les articulacions que s'ha estimat.

de solucionar l'equació diferencial (4.8). Per altra banda, en aquest codi s'aproxima amb *Fourier* en base cosinus (4.3).

$$s_j = \exp\left(-\frac{\alpha}{\tau}t_j\right) \quad (4.1)$$

$$f_{target}(s_j) = \frac{\tau\dot{v}_j + Dv_j}{K} - (g - x_j) + (g - x_0)s_j \quad (4.2)$$

$$\psi_i(s_j) = \cos(\pi n s_j) \quad (4.3)$$

L'aproximació és per mínims quadrats, prenent les matrius de l'equació (4.4) per tal d'extreure els valors de  $w_i$ . Aquests pesos es calculen resolent el sistema matricial (4.5).

$$A_{m,n} = \begin{pmatrix} \psi_1(s_1) & \psi_2(s_1) & \cdots & \psi_n(s_1) \\ \psi_1(s_2) & \psi_2(s_2) & \cdots & \psi_n(s_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(s_m) & \psi_2(s_m) & \cdots & \psi_n(s_m) \end{pmatrix} \quad W_m = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} \quad Y_m = \begin{pmatrix} f(s_1) \\ f(s_2) \\ \vdots \\ f(s_m) \end{pmatrix} \quad (4.4)$$

$$(A_{m,n}^T A_{m,n})W_m = A_{m,n}^T Y_m \quad (4.5)$$

## Càlcul de les accions de control

A l'hora de dur a terme el control del robot, s'han d'utilitzar els pesos  $w_i$  calculats anteriorment i tan sols depenent del valor de  $s$  de l'equació (4.8) i del *transformation system* de les *DMPs* expressat per les equacions (4.6) i (4.7).

$$\tau\dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) \quad (4.6)$$

$$\tau\dot{x} = v \quad (4.7)$$

$$\tau\dot{s} = -\alpha s \quad (4.8)$$

Aquest codi crea les *DMPs* sense problemes, com s'ha es veu en [40] que també l'utilitza. Ara bé, té el problema que no porta implementat cap tipus d'algorisme de millora de la política a partir de paràmetres externs a la pròpia dinàmica del sistema. Per això, s'ha preferit utilitzar el **package** que s'explica a continuació.

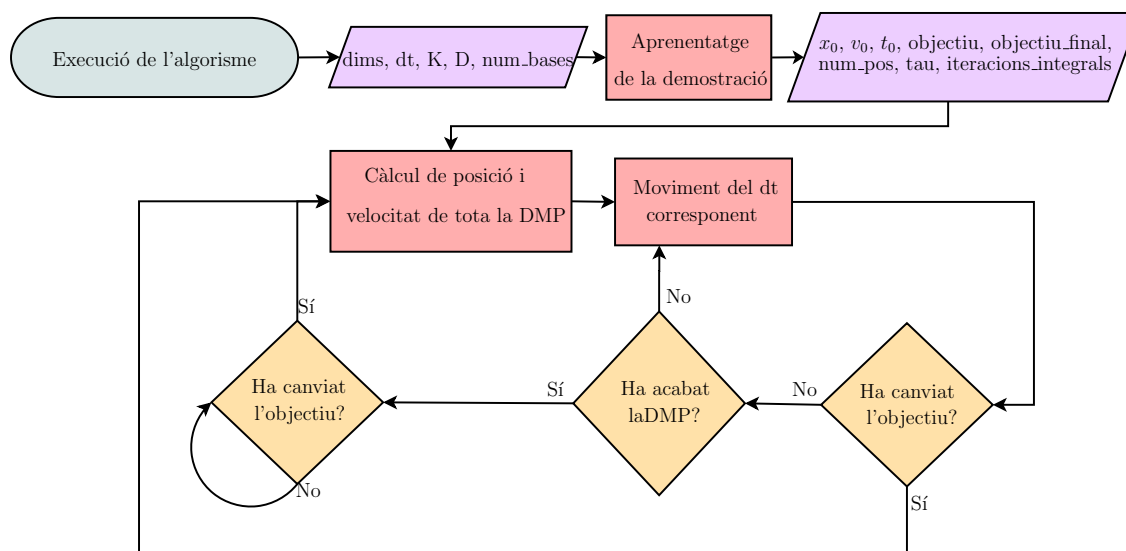


Figura 4.1: Diagrama de fluxos del suposat funcionament en aquest cas del codi de Scott Niekum.

## 2.4.2 Package complet del robot PR2 del *USC-CLMC*

A continuació es presenta un conjunt de **packages** públics, es poden trobar en aquesta web [38] del *USC-CLMC* (*Computational Learning and Motor Control Lab at the University of Southern California*). Aquests permeten l'ús de les DMPs amb l'algorisme  $PI^2$ , en el robot PR2 de Willow Garage [6], incloent el control en temps real per interfície gràfica, l'ús de percepció auditiva, etc. Per tant, ha hagut de ser adaptat al cas del treball, enfocat per l'AIBO, amb molts menys complements. Per fer-ho, s'han eliminat molts dels **packages** que s'hi inclouen, perquè no s'han d'utilitzar. A més, s'ha hagut de crear un codi d'execució, perquè realment el **package** és la base sobre la que pot funcionar l'algorisme, però es necessita un codi que l'engegui.

El fet d'eliminar els **package** pot semblar una tasca fàcil, però al darrera hi ha hagut un gran treball d'entendre un conjunt de codis, creats per altres persones i, a més, de gran complexitat<sup>1</sup>. Per altra banda, a mesura que s'ha avançat en el coneixement del conjunt del **package** i en la creació del codi d'execució, s'ha anat modificant el conjunt per permetre el funcionament global. En aquest apartat, inicialment s'explica els trets més importants dels **packages** que finalment s'han utilitzat, i cap el final s'esmenten alguns dels canvis respecte el codi original i el perquè d'aquests.

<sup>1</sup>Es considera de gran complexitat tant per la grandària, com per estar pensat per usuaris de coneixements avançats en el llenguatge de programació *C++*.

## DMP

El **package** de les *DMPs* és el nucli de tot l'algorisme de control. En la seva base, el **subpackage** `dynamic_movement_primitive` juntament amb `locally_weighted_regression`, funcionen de la mateixa forma que el codi presentat anteriorment de Scott Niekum. Ara bé, canvien dos aspectes respecte l'anterior: (1) treballa amb funcions gaussianes, del tipus  $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ , en lloc d'utilitzar *Fourier* (4.3); (2) la comunicació amb el propi codi és fa des de *ROS*.

El **subpackage** `dynamic_movement_primitive` s'encarrega del càlcul de les accions de control i de concedir la comunicació de *ROS* al conjunt. Amb aquest tipus de comunicació es permet modificar els paràmetres de la *DMP* sense la necessitat de modificar el codi, sinó que és pot fer modificant les propietats del node d'execució<sup>2</sup>. En el codi d'execució d'aquest treball, aquests paràmetres estan donats de forma automàtica dins el codi. Altres grans avantatges de la comunicació mitjançant *ROS* són el poder publicar l'estat de la *DMP*, en tot moment, a través d'un **topic** i la facilitat d'intercanviar informació amb l'exterior.

Per altra banda, el **subpackage** `locally_weighted_regression` és el que s'encarrega de la transformació la trajectòria programada de mínim *jerk*<sup>3</sup> en la suma del sistema dinàmic amb el conjunt de gaussianes, explicat a l'apartat 2.3.4. Tot aquest procés és requereix perquè la *DMP* de cada articulació pugui ser creada. La classe principal és *LWR*, que permet definir el conjunt de gaussianes, aquesta és característica per cada articulació. Tot i ser un codi diferent al de Niekum, realment els dos duen a terme els mateixos càlculs, però expressats de forma diferent i utilitzant gaussianes.

## Policy Learning

Fins aquest punt s'ha explicat la base de les *DMPs*, sense qualsevol de les parts anteriors no seria possible dur-les a terme. Ara bé, la integració de l'algorisme  $PI^2$  és, en realitat, un complement, per millorar-ne l'eficàcia. Aquest **package** integra  $PI^2$ , juntament amb varies derivacions d'aquest, que poden ser utilitzats en altres casos.

La interacció entre aquest i les *DMPs* és produeix a partir d'haver-se fet l'aproximació per gaussianes. En aquest moment, comença el procés iteratiu exemplificat en la figura 4.2, on es modifiquen els pesos per tal de minimitzar una funció objectiu. Per tant, si la funció objectiu del  $PI^2$  és l'adequada, el moviment dut a terme pel robot millora. Si

<sup>2</sup>Això és pot fer amb la comanda `node_handle.setParam("nom_paràmetre", valor_paràmetre)` o amb `usc_utilities::write(node, "nom_paràmetre", valor_paràmetre)` si és un vector.

<sup>3</sup>Aquesta trajectòria és creada pel propi **subpackage** `dynamic_movement_primitive`.

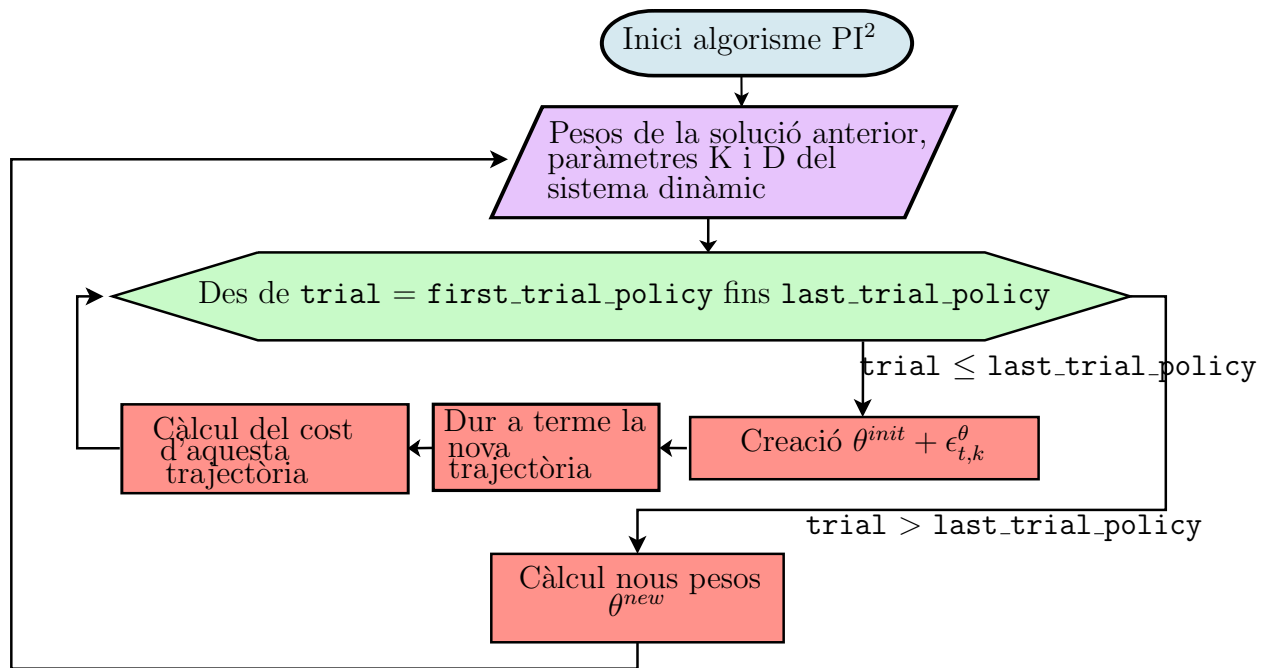


Figura 4.2: Procés iteratiu del codi Policy\_learning

es desitja modificar la funció objectiu s'ha d'accedir a l'arxiu `dmp_waypoint_task.cpp` al subpackage `policy_improvement.loop`.

## Capítol 3

# Disseny

Inicialment, s'ha plantejat un disseny amb un model del robot que serviria per simular tot el conjunt amb l'ordinador. D'aquí en sortiria un solució inicial per implementar al robot físic, i a partir d'aquest arribar a la solució final. La solució primera estalviaria possibles perills, tant pel propi robot com per l'entorn, i ajudaria a aconseguir la convergència a una solució final més ràpidament. En referència a les *DMPs*, com s'ha explicat en l'apartat 2.4.2, el codi utilitzat és el creat per *USC-CLMC*, però modificat per l'adaptació a aquest treball. Ara bé, aquest dona dues opcions d'execució a escollir segons l'algorisme: (1) *DMPs* conjuntament amb l'algorisme  $PI^2$ , (2) *DMPs* sense el  $PI^2$ ; el funcionament dels dos tipus es troben explicats en aquest mateix apartat. Per altra banda, en aquest apartat també s'explica l'ús del conjunt acceleròmetre-giroscopi, en l'algorisme de control, i de influència i control de la plataforma.

### 3.1 Model de l'AIBO

Com s'ha explicat en l'apartat 1.6.4, hi ha una gran multitud de possibles formes de fer un model per un robot. Ara bé, tots aquests mètodes són complexos i requereixen de bastant temps. Per això, s'ha considerat que fer el model seguint algun d'aquests mètodes no es trobava dins l'abast del treball i, per tant, s'ha escollit fer el model escrivint el codi de forma manual. Per dur a terme el model s'ha pres com a referència les mesures aproximades de la figura 1.1.

A l'hora de fer el model es poden escollir diferents formats d'arxiu, tals com `urdf`, `model` o `sdf`. Ara bé, d'aquests l'únic que tan és compatible amb `rviz`<sup>1</sup> i com amb `Gazebo`<sup>2</sup>,

---

<sup>1</sup>Mencionat en l'apartat 1.6.2.

<sup>2</sup>Mencionat en l'apartat 1.6.4.

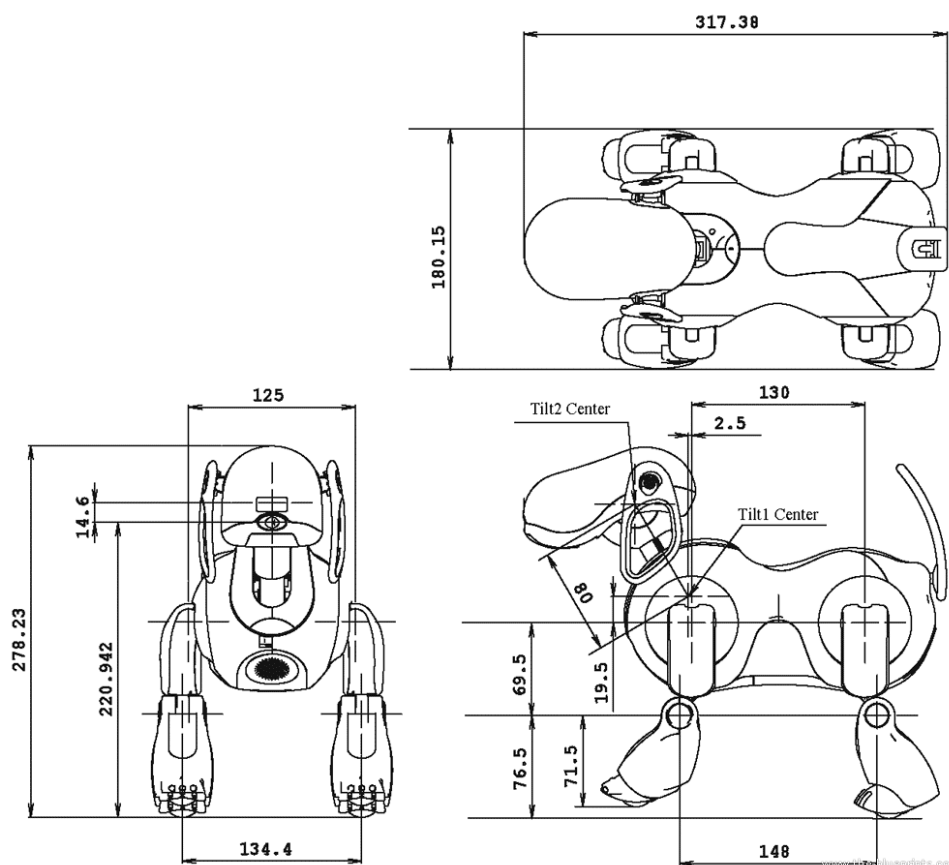


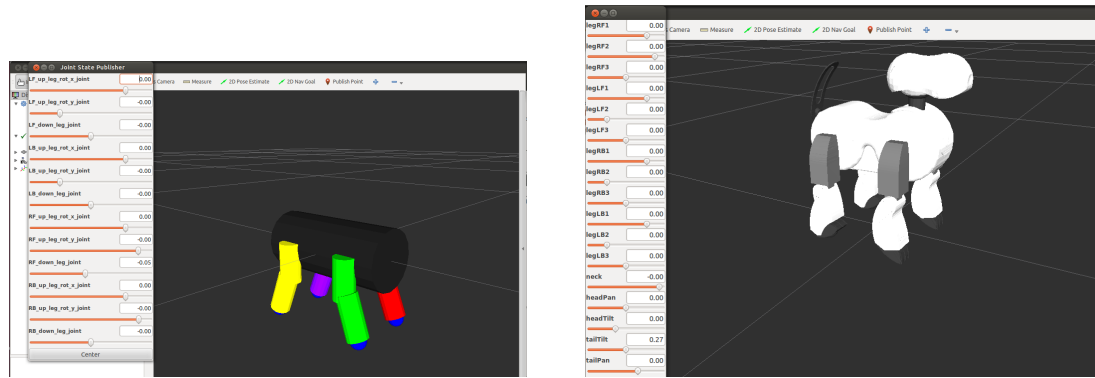
Figura 1.1: Mesures en *mm* de l'AIBO [5]

pel control de robots mòbils, és l'*urdf*. Tot i que amb aquest format al ser utilitzat en *Gazebo* existeix un petit retard a diferència del *sdf*, que és el format òptim, s'ha escollit per fer el model en *urdf* [56].

El model inicial que s'ha fet consta de nou cilindres, un pel cos, quatre per les parts superiors de les potes i quatre més per les parts inferiors. A més de quatre esferes pels punts de contacte amb el terra. Les unions entre les potes o entre el cos i les potes són de tipus *revolute*, aquest tipus defineix una articulació que gira entorn de l'eix indicat i en la posició donada. Per altra banda, els punts de contacte al terra són fixos i, a més, amb fricció molt alta<sup>3</sup>. Un fet a tenir en compte en aquest model és que no s'ha tingut en compte la inèrcia de cadascuna de les parts del robot, deguda al temps que requeriria fer el càlculs.

En un model es poden diferenciar dues estructures molt clares. Per una banda, existeix la part física, que és la que es té en compte a l'hora de fer les col·lisions, i en general tots els càlculs. Per altra banda, està la part visual que és la decorativa, la veu l'usuari

<sup>3</sup>La fricció alta ha sigut posada perquè es pugui provar el model en *Gazebo*, tot i que realment en el model final no s'ha de posar d'aquesta manera.



(a) Model inicial de l'AIBO en l'entorn **Rviz** i amb el control de les articulacions (esquerra). (b) Model final de l'AIBO en l'entorn **Rviz** i amb el control de les articulacions (esquerra).

Figura 1.2: Models de l'AIBO en l'entorn **rviz**

en simular el robot. La idea per fer en el model final ha estat recrear aquest tant amb la part física com visual que té el robot, de tal manera que els càlculs i la visualització siguin molt més realistes. Per dur a terme aquesta tasca s'han hagut de descarregar les textures de l'AIBO de tipus **Sketchup** de [53], prendre cada part del robot per separat i guardar-les en arxius separats.

Finalment, s'ha deixat la continuïtat de la construcció del model, quedant-se amb el model inicial, tot i haver fet algunes proves d'introduir les textures de l'AIBO. La causa és que s'ha vist que el benefici que comporta la creació d'un model, amb els algorismes que s'estan plantejant a utilitzar, no és rendible pel temps que s'hi ha de dedicar. En gran part és degut a la no necessitat d'un model per part dels algorismes, ja que poden treballar perfectament sense aquest.

Ara bé, el projecte del model es continuat, en l'estat que s'ha deixat en aquest treball (model inicial amb control de les articulacions, textures classificades i preparades per ser implementades), per l'estudiant Diego Muñoz, veure figura 1.2b. Aquest l'utilitza per veure l'estat del robot físic a través de l'aplicació **rviz**. A més, el model està també en el Github del *TFG*<sup>4</sup>, amb el nom d'*aibo\_model*, i pot ser utilitzat i millorat per qualsevol persona que s'hi vulgui dedicar.

## 3.2 Moviment del centre de gravetat

Tal i com s'ha vist al llarg del treball, uns dels requisits per poder aconseguir recuperar l'estabilitat és tenir les mesures dels moviments del robot. Com a idea inicial, s'ha pensat

<sup>4</sup><https://github.com/lluissalord/TFG/>



utilitzar el sensor triaxial d'accelerometria que porta incorporat el propi AIBO, i d'aquesta forma poder estudiar el moviment d'aquest. Però abans s'han fet unes comprovacions de la fiabilitat de les dades d'aquest sensor. Aquestes comprovacions es basen en l'observació de la variabilitat de les mesures i comprovar si l'angle d'inclinació extret a partir de l'acceleració és el mateix que el mesurat externament. Finalment ha resultat que no és gens precís per la tasca que ha de desenvolupar.

### 3.2.1 Acceleròmetre MPU6050 i giroscopi GY-521

Al no ser el sensor del propi robot d'utilitat en el treball, s'ha decidit comprar un altre sensor triaxial d'accelerometria amb un giroscopi de tres eixos. En aquest cas, s'ha optat per una peça on hi ha integrat l'acceleròmetre MPU6050 i el giroscopi GY-521. S'ha escollit aquest, en primera instància, perquè està enfocat per a ser utilitzat amb Arduino, que és un microcontrolador del que es disposa en el departament i que ja es pensa utilitzar per altres motius. A més, existeix molta informació per la xarxa, com llibreries<sup>1</sup> o aplicacions fetes, a banda de ser recomanat per la seva qualitat-preu.

Ara bé, el fet de formar part de l'AIBO crea la problemàtica d'haver d'implementar una comunicació entre el sensor i l'algorisme de control. Com que és té un entorn de treball, *ROS*, que ja s'està utilitzant per comunicar-se amb l'ordinador, és pot aprofitar aquest entorn per tal d'incorporar la informació que es vulgui enviar al robot.

A partir de l'acceleròmetre es treu informació de les acceleracions que pateix aquest sensor, incloent la gravetat, aquest transmet les dades en RAW que s'han de dividir per 16384 per a ser transformats a  $Gs^2$ . Per altra banda, el giroscopi mesura la velocitat angular, abans d'utilitzar les dades que arriben directament d'aquest, s'ha de passar del RAW a  $^\circ/s$  (graus per segon) dividint-ho 131.

### 3.2.2 Càlcul de la posició i de la inclinació

Com s'ha explicat en l'apartat 2.2 d'Estabilitat, una de les millors formes de cercar la màxima estabilitat, dins l'abast del *TFG*, és provocar que el *CdG* retorni a la posició d'origen. Per tant, es requereix saber en tot moment quan s'ha desplaçat respecte aquest origen, però per això es necessita tenir la informació de la inclinació del robot, per tal d'extreure tan sols la component de l'acceleració que ens interessa.

<sup>1</sup>Entre d'elles un de les més utilitzades i molt pràctica és la *i2cdevlib*, on a banda de per aquest conjunt acceleròmetre-giroscopi, també n'hi ha per molts altres [49]

<sup>2</sup>Força  $G$ , per tant, el resultat de la divisió s'ha de multiplicar per 9,80665 per passar-ho a  $m/s^2$ .

## Càlcul de la inclinació

En aquest càlcul, es requereix essencialment del giroscopi, perquè tant sols s'han d'integrar les dades subministrades per aquest. L'acceleròmetre només podria servir, per extreure la inclinació, si s'estigués estàtic o, com és en aquest cas, per ajudar a filtrar les dades. Per tal de dur a terme aquest filtre, s'han trobat dos tipus de filtres que podrien ser útils, el filtre **Kalman** i el filtre **complementari**.

El filtre Kalman es basa en l'estimació de l'estat del sistema a partir de la informació de l'acceleròmetre i el giroscopi, en el moment anterior i en l'actual. Per altra banda, el filtre complementari és basa en l'ús de les dades del giroscopi, per la mesura en temps curts, i realitzar la correcció de la deriva, amb les dades de l'acceleròmetre [12]. Un filtre complementari és un filtre Kalman, però en unes condicions i restriccions determinades. Per això no difereixen gaire en els resultats donats, el Kalman és més precís, però requereix de més recursos computacionals.

Inicialment, s'ha plantejat utilitzar el filtre Kalman, per això s'ha pres el codi de [26], on està l'algorisme del Kalman i del filtre complementari fets en un exemple per Arduino. El problema ha sorgit en intentar implementar tot el conjunt de filtre Kalman i les llibreries que permeten la comunicació de l'*Arduino Uno* amb *ROS*, perquè la placa no té prou memòria per abarcar-ho tot. Pel que sembla, la llibreria de *ROS* ocupa pràcticament la meitat de la memòria del microcontrolador i la resta no és suficient per l'algorisme del filtre Kalman, tot i haver intentat optimitzar al màxim l'ús de memòria.

Una opció per solucionar el problema anterior hagués pogut ser transmetre per *ROS* tan sols les dades RAW dels sensors, i llavors fer el filtratge en l'ordinador, però s'hauria de tenir un procés més en paral·lel. Però s'ha cregut més convenient utilitzar el filtre complementari, que com s'ha dit requereix de menys recursos computacionals i s'ha provat que si que pot implementar tot el conjunt.

El filtre complementari matemàticament és molt simple (2.1), tant sols depèn d'un paràmetre  $\lambda$  que determina quin percentatge es pren de l'angle calculat amb l'acceleració i la resta pel giroscopi. Ara bé, per tal de poder ser útil s'hi han hagut de fer algunes millores:

- Les dades que s'utilitzen són una mitja de quinze mostres RAW
- Només varia la publicació a *ROS* si la diferència entre l'angle anterior i l'actual és major a  $0,02^\circ$

$$\theta_{compl} = (1 - \lambda)(\theta_{compl} + w_{giro}dt) + \lambda\theta_{accel} \quad (2.1)$$

### **3.3    *Goals* de les *DMPs* i funció recompensa**

### **3.4    Plataforma**

## Capítol 4

# Funcionament i execució

En aquest apartat es pensa explicar quins són els passos que s'han de seguir per tal d'aconseguir dur a terme l'execució del conjunt.

### 4.1 Connexions a establir

Un dels aspectes que fins aquest instant no s'ha esmentat és la configuració de les connexions. Per tal de poder transmetre tota la informació que requereix l'algorisme de control s'ha de donar un canal de comunicació amb el conjunt acceleròmetre-giroscopi, com també un altre amb les articulacions del propi AIBO.

#### 4.1.1 Conjunt acceleròmetre-giroscopi

En l'apartat 2.1.3 s'ha explicat per sobre com es transmet la informació des de l'Arduino a *ROS*. Aquí s'exposa la comunicació entre el sensor i el microcontrolador. Aquesta es basa en el bus de comunicació I<sup>2</sup>C, la principal característica és que utilitza dues línies, una per la informació **SDA** i una per la senyal de rellotge **SCL**, a banda d'una línia de terra.

L'I<sup>2</sup>C és un bus de comunicació bidireccional, però no permet enviar i rebre informació de forma simultànea. El seu funcionament comença i acaba amb unes combinacions determinades dels nivells de la línia **SDA** i **SCL**. A l'inici de la transmissió, s'envia la direcció<sup>1</sup> del dispositiu al que va dirigida la informació amb un indicador de lectura o escriptura. A més, aquest bus després de cada **byte** es valida amb un bit de reconeixement **ACK**

---

<sup>1</sup>Aquesta direcció és única per cada dispositiu connectat

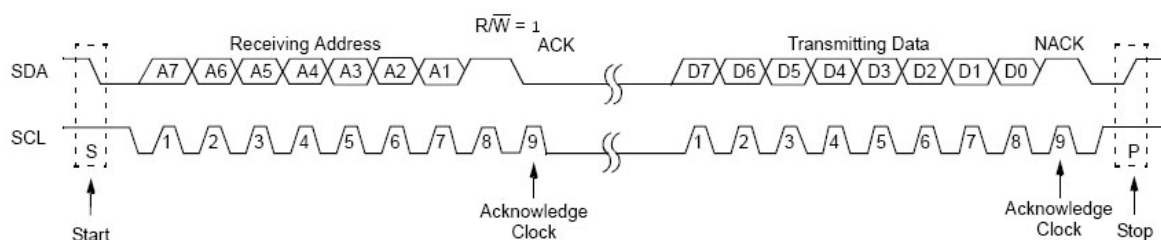
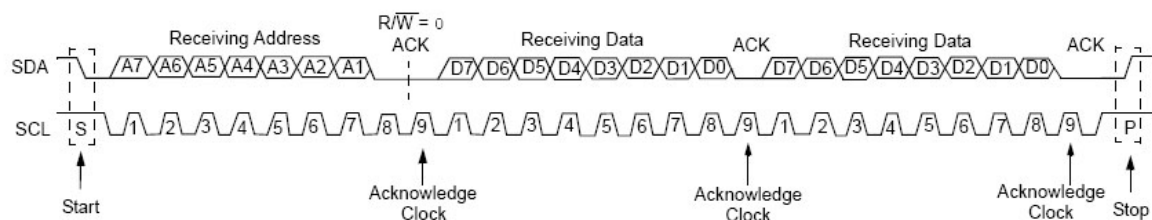


Figura 1.1: Esquemes del funcionament del bus I<sup>2</sup>C d'escriptura (dalt) i de lectura (baix).

(*Acknowledge Clock*). Tot aquest procés es pot veure exemplificat tant per escriptura, figura 1.1a, com per lectura, figura 1.1b.

En el cas del sensor és fàcilment distingir cadascun dels pins corresponents, ja que els noms corresponen amb la nomenclatura de la placa. Ara bé, en el microcontrolador Arduino Uno els pins **SDA** i **SCL** no es troben designats sobre aquesta placa, tot i que s'hagin d'utilitzar uns en concret. Aquests pins són el **A4** pel **SDA** i el **A5** pel **SCL**. Per últim, cal a dir que l'alimentació del conjunt acceleròmetre-giroscopi ha de ser la de 3 V, per tant, les connexions quedarien com es veu en la figura 1.2.

#### 4.1.2 AIBO a través d'*Access Point* (AP)

Si es desitja establir una comunicació entre l'algorisme de control i l'AIBO de forma continua, no hi ha altra solució que fer-ho amb una connexió sense fils. Ara bé, hi ha dos modes de connexió: (1) a través d'*Access Point*, (2) sense *Access Point*. En aquest treball s'ha utilitzat la primera, ja que en el moment de fer les proves l'altre mètode ha donat problemes per establir la connexió.

Abans de començar amb la descripció del procediment de connexió amb  $AP$ , s'ha de tenir en compte que el tipus de xifratge de la contrasenya ha de ser **WEP** o sense contrasenya. Per altra banda, en aquest apartat es suposen uns coneixements bàsics en nomenclatura de protocols TCP/IP.

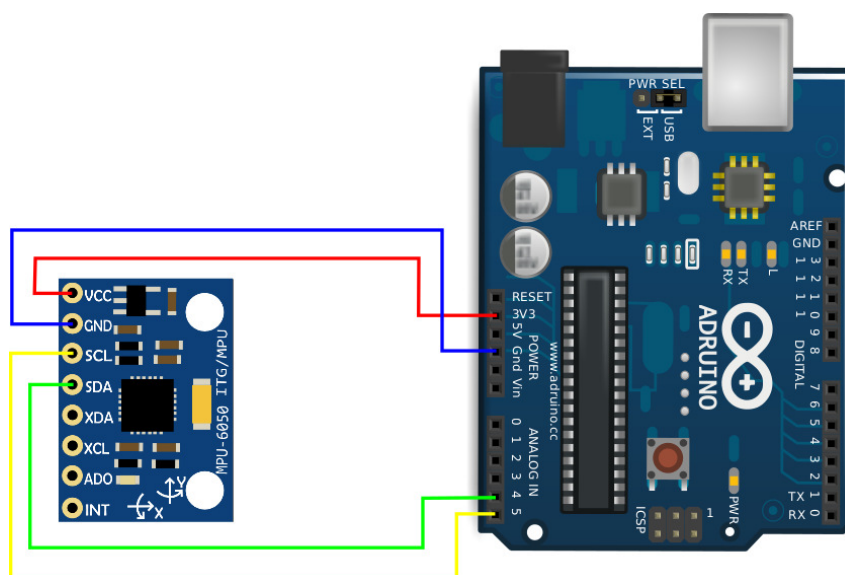


Figura 1.2: Connexions del conjunt acceleròmetre-giroscopi amb l'Arduino.

Per tal de configurar la targeta de xarxa de l'AIBO s'ha d'accedir al fitxer de configuració de la targeta de memòria `OPEN-R\SYSTEM\CONF\WLANDFLT.TXT`. Una possible configuració és la que s'exposa a continuació, suposant que la porta d'enllaç predeterminada és la 192.168.10.1, la mascara de subred és 255.255.255.0, canal de connexió numero 3 i amb AIBONET i AIBO2 com a nom de xarxa i contrasenya, respectivament. Si és desitja informació més detallada o fer la connexió sense *Acces Point* es pot trobar en [59].

```

HOSTNAME=AIBO
ETHER_IP=192.168.10.124
ETHER_NETMASK=255.255.255.0
IP_GATEWAY=192.168.10.1
ESSID=AIBONET
WEPENABLE=1
WEPKEY=AIBO2
APMODE=2 # this mode indicates auto-mode
CHANNEL=3

```

Havent configurat la targeta de xarxa, en engegar-se l'AIBO busca la xarxa del nom que se li ha donat i s'hi connecta. Finalment, tan sols quedaria crear la comunicació *ROS*, que es crea a partir d'un script, explicat en el següent apartat, però que abans requereix que l'AIBO estigui en la mateixa xarxa que l'algorisme.

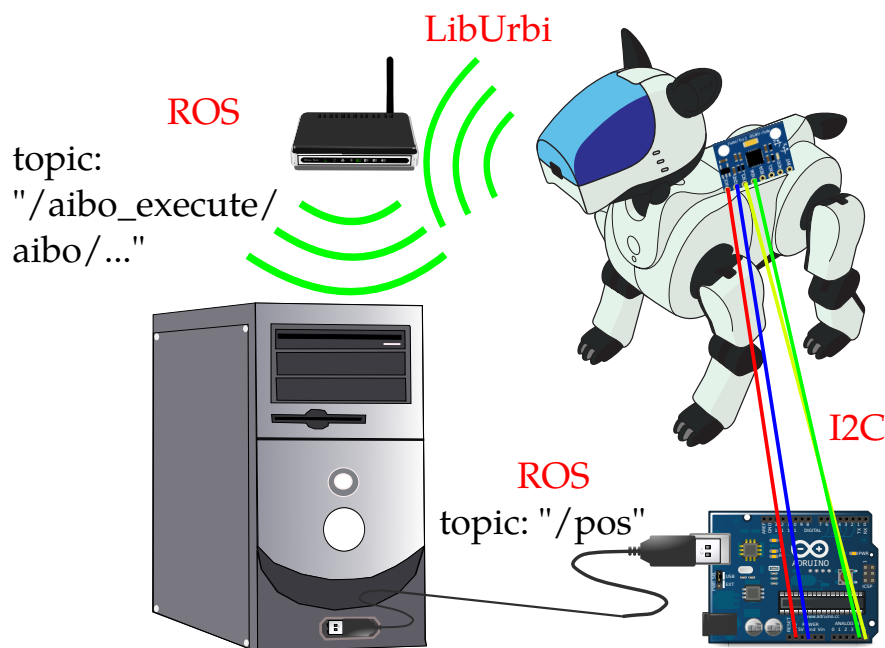


Figura 1.3: Esquema de connexions entre AIBO-ROS i sensor-Arduino-ROS

## 4.2 Processos previs

En primer terme, pel bon funcionament de les *DMPs* cal engegar una sèrie de processos per permetre l'intercanvi d'informació i per dur a terme el control de la plataforma, si escau. En aquest apartat s'explica com s'ha de procedir per posar en funcionament tot el conjunt. Ara bé, es dona per suposat que el *software* requerit ja s'ha instal·lat<sup>1</sup>.

### 4.2.1 Carrega del programa de control del *CdG* a Arduino

En el Github del *TFG*<sup>2</sup> existeix un programa d'Arduino (`arduino/MPU6050_compl.ino`) que ha de ser arreglat a la placa per tal de poder transmetre de forma correcta la informació al topic de *ROS* corresponent (`/pos`).

Per poder carregar el programa s'ha de tenir connectat l'Arduino amb un cable USB a l'ordinador. Després també s'ha d'obrir, amb l'IDE d'Arduino, el fitxer nombrat anteriorment, per finalment prémer a l'opció **Archivo->Subir**.

<sup>1</sup>El procés d'instal·lació està explicat en els Annexos.

<sup>2</sup><https://github.com/lluissalord/TFG>

### 4.2.2 Execucions per línia de comandes

Cadascuna de les accions següents s'ha de realitzar per separat en un terminal independent, preferiblement en aquest ordre per evitar possibles falls i perills:

**\$ roscore** executa el *ROS Master*<sup>3</sup>

**\$ rosrun rosserial\_python serial\_node.py \dev\ttyACM0** activa la comunicació entre l'Arduino i *ROS*<sup>4</sup>. Pot ser que en algun moment l'entrada de dades no sigui \dev\ttyACM0, per saber-ho s'ha d'observa en el IDE d'Arduino a quin port està connectat.

**\$ rosrun aibo\_server aibo\_server 192.168.0.124** execució del programa que permet la comunicació AIBO-*ROS*. El número 192.168.0.124 és la IP de l'AIBO, varia segons la configuració que s'ha seguit en l'apartat 4.1.2.

**\$ rosrun dmp\_execute dmp\_execute 0** engega l'algorisme de control. El número final varia segons 1 o 0, segons si es desitja utilitzar el PI<sup>2</sup> o no, respectivament.

## 4.3 Funcionament de *DMPs* sense PI<sup>2</sup>

## 4.4 Funcionament de *DMPs* amb PI<sup>2</sup>

---

<sup>3</sup>Concepte explicat en l'apartat 1.6.2

<sup>4</sup>Primer s'ha de carregar el programa, explicat en l'apartat 4.2.1, a la l'Arduino, si la placa no el té carregat.



## Capítol 5

## Conclusions

# Agraïments



org/EBooks/Computers/ArtificialIntelligence/Cord\_Cunningham-Machine\_Learning\_Techniques\_for\_Multimedia-9783540751700.pdf.

- [9] Archi Delphinanto, Ton Koonen, and Frank den Hartog. End-to-end available bandwidth probing in heterogeneous IP home networks. *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 431–435, 2011. doi: 10.1109/CCNC.2011.5766506.
- [10] Institut d’Estudis Catalans. Diccionari de la llengua catalana. URL <http://dlc.iec.cat/index.html>.
- [11] Masahiro Fujita. Digital creatures for future entertainment robotics. *Robotics and Automation, 2000. Proceedings. ICRA’ ...*, (April), 2000. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=844149](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=844149).
- [12] David Gaydou, J Redolfi, and A Henze. Filtro complementario para estimacion de actitud aplicado al controlador embebido de un cuatrirrotor. *...de Sist. Embebidos*, 2011. URL [http://proyectos.ciii.frc.utn.edu.ar/cuadricoptero/export/9ed95816c90cc7d83e32fd2e13b032dc515c0d7a/documentacion/informe\\_final/paper\\_case.pdf](http://proyectos.ciii.frc.utn.edu.ar/cuadricoptero/export/9ed95816c90cc7d83e32fd2e13b032dc515c0d7a/documentacion/informe_final/paper_case.pdf).
- [13] Gostai. URBI Doc for Aibo ERS2xx ERS7 and URBI 1.0. URL <http://www.gostai.com/doc/en/aibo/>.
- [14] Philipp Hennig. Optimal Reinforcement Learning for Gaussian Systems. *NIPS*, pages 1–9, 2011. URL <https://papers.nips.cc/paper/4410-optimal-reinforcement-learning-for-gaussian-systems.pdf>.
- [15] Lukas Hohl, Ricardo Tellez, Olivier Michel, and Auke Jan Ijspeert. Aibo and Webots: Simulation, wireless remote control and controller transfer. *Robotics and Autonomous Systems*, 54(6):472–485, June 2006. ISSN 09218890. doi: 10.1016/j.robot.2006.02.006. URL <http://linkinghub.elsevier.com/retrieve/pii/S0921889006000327>.
- [16] HONDA. History of ASIMO Robotics — ASIMO Innovations by Honda. URL <http://asimo.honda.com/asimo-history/>.
- [17] Vincent Hugel and P Blazevic. Towards efficient implementation of quadruped gaits with duty factor of 0.75. *Robotics and Automation, 1999. ...*, (May), 1999. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=770458](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=770458).
- [18] International Organization for Standaritzacion. ISO 8373:2012: Robots and robotic devices — Vocabulary. Technical report, ISO, Genève, 2012.

- [19] Ion Ion, Ion Simionescu, and Marius Ungureanu. Stability Analysis of Gaits of Quadruped Walking Robot MERO. ... *Workshop on Mobile Robots*, .... URL <http://www.profesaulosuna.com/data/files/ROBOTICA/ROBOT/20.pdf>.
- [20] LP Kaelbling, ML Littman, and AW Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 4:237–285, 1996. URL <http://arxiv.org/abs/cs/9605103>.
- [21] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258, November 2010. ISSN 0278-3649. doi: 10.1177/0278364910388677. URL <http://ijr.sagepub.com/cgi/doi/10.1177/0278364910388677>.
- [22] Oussama Khatib, Oliver Brock, Kyong-Sok Chang, Francois Conti, Diego Ruspini, and Luis Sentis. Robotics and interactive simulation, 2002. ISSN 00010782.
- [23] Jens Kober and Jan Peters. Reinforcement learning in robotics: A survey. *Reinforcement Learning*, 2012. URL [http://link.springer.com/chapter/10.1007/978-3-642-27645-3\\_18](http://link.springer.com/chapter/10.1007/978-3-642-27645-3_18).
- [24] Adam Daniel Laud. *Theory and Application of Reward Shaping in Reinforcement Learning*. PhD thesis, University of Illinois at Urbana-Champaign, 2004. URL <https://www.ideals.illinois.edu/bitstream/handle/2142/10797/TheoryandApplicationofRewardShapinginReinforcementLearning.pdf?sequence=2>.
- [25] Tim Laue, Kai Spiess, and T Röfer. SimRobot—a general physical robot simulator and its application in robocup. *RoboCup 2005: Robot Soccer World Cup IX*, pages 173–183, 2006. URL [http://link.springer.com/chapter/10.1007/11780519\\_16](http://link.springer.com/chapter/10.1007/11780519_16).
- [26] Kristian Lauszus. KalmanFilter - TKJElectronics, 2012. URL <https://github.com/TKJElectronics/KalmanFilter>.
- [27] Erik G. Learned-Miller. *Introduction to Supervised Learning*. PhD thesis, University of Massachusetts, Amherst, 2014. URL <http://people.cs.umass.edu/~elm/Teaching/Docs/supervised2014a.pdf>.
- [28] TT Lee, CM Liao, and TK Chen. On the stability properties of hexapod tripod gait. *Robotics and Automation, IEEE ...*, 4(4), 1988. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=808](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=808).
- [29] Vo-Gia Loc, Se-goh Roh, Ig Mo Koo, Duc Trong Tran, Ho Moon Kim, Hyungpil Moon, and Hyouk Ryeol Choi. Sensing and gait planning of quadruped walking and

- climbing robot for traversing in complex environment. *Robotics and Autonomous Systems*, 58(5):666–675, May 2010. ISSN 09218890. doi: 10.1016/j.robot.2009.11.007. URL <http://linkinghub.elsevier.com/retrieve/pii/S0921889009002048>.
- [30] Vo-Gia Loc, Ig Mo Koo, Duc Trong Tran, Sangdoek Park, Hyungpil Moon, and Hyouk Ryeol Choi. Improving traversability of quadruped walking robots using body movement in 3D rough terrains. *Robotics and Autonomous Systems*, 59(12):1036–1048, December 2011. ISSN 09218890. doi: 10.1016/j.robot.2011.08.007. URL <http://linkinghub.elsevier.com/retrieve/pii/S0921889011001588>.
- [31] Ignacio Pedrosa Lojo. *Proyecto MIRHO (Mobile Intelligent Hexapod Robot)*. PhD thesis, Universitat Politècnica de Catalunya, 2009. URL <http://upcommons.upc.edu/handle/2099.1/6488>.
- [32] R Menéndez Paredes. *Control y supervisión inalámbrica de la plataforma robótica Pleo*. PhD thesis, Universitat Politècnica de Catalunya, 2011. URL <http://upcommons.upc.edu/handle/2099.1/11801>.
- [33] Olivier Michel. Cyberbotics Ltd. Webots: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1:40–43, 2004.
- [34] Bilge Mutlu, S. Osman, Jodi Forlizzi, J. Hodgins, and S. Kiesler. Perceptions of ASIMO: an exploration on co-operation and competition with humans and humanoid robots. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 351–352. ACM, 2006. ISBN 1595932941. doi: 10.1145/1121241.1121311. URL <http://portal.acm.org/citation.cfm?id=1121311>.
- [35] K Nagasaka, Y Kuroki, and S Suzuki. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. *Robotics and ...*, pages 3189–3194, 2004. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1308745](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1308745).
- [36] Duy Nguyen-Tuong and Jan Peters. *Model learning for robot control: a survey.*, volume 12. November 2011. ISBN 3405062780. doi: 10.1007/s10339-011-0404-1. URL <http://www.ncbi.nlm.nih.gov/pubmed/21487784>.
- [37] Kei Okada. Roseus - ROS package. URL <http://wiki.ros.org/roseus>.
- [38] Peter Pastor and Mrinal Kalakrishnan. ROS packages from the USC-CLMC. URL <https://github.com/usc-clmc/usc-clmc-ros-pkg>.
- [39] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, May 2009.

- doi: 10.1109/ROBOT.2009.5152385. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5152385>.
- [40] Sammy Pfeiffer. *Gesture learning and generation and execution of slightly modified gestures*. PhD thesis, Universitat Politècnica de Catalunya, 2014.
- [41] A. Purushotham and G. Venkata Rao. Dynamic stability analysis of a quadruped robotic manipulator system: analytical approach. *International Journal of Applied Engineering Research*, 2009.
- [42] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software*, 2009.
- [43] Marc Raibert and Kevin Blankespoor. Bigdog, the rough-terrain quadruped robot. *Proceedings of the 17th IFAC World Congress*, 2008. URL [http://web.unair.ac.id/admin/file/f\\_7773\\_bigdog.pdf](http://web.unair.ac.id/admin/file/f_7773_bigdog.pdf).
- [44] Rainersen. Imatges Aibo ERS-7. URL <http://rainersen.de/aibo/>.
- [45] Carlos Mario Ramos Olave. Diseño, implementación y control visual de posición de un sistema placa-bola. 2013.
- [46] Robot Operating System. ROS.org. URL <http://www.ros.org/core-components/>.
- [47] PAL Robotics. REEM-C. URL <http://pal-robotics.com/en/robots/reem-c>.
- [48] T Röfer and Tim Laue. On B-Human's code releases in the Standard Platform League-software architecture and impact. 2013. URL <http://www.informatik.uni-bremen.de/kogrob/papers/RC-Roefer-Laue-14.pdf>.
- [49] Jeff Rowberg. I2C Device Library. URL <http://www.i2cdevlib.com/>.
- [50] Elmar Rückert and Andrea D'Avella. Learned parametrized dynamic movement primitives with shared synergies for controlling robotic and musculoskeletal systems. *Frontiers in computational neuroscience*, 7(October):138, January 2013. ISSN 1662-5188. doi: 10.3389/fncom.2013.00138. URL <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3797962&tool=pmcentrez&rendertype=abstract>.
- [51] Elmar a Rückert, Gerhard Neumann, Marc Toussaint, and Wolfgang Maass. Learned graphical models for probabilistic planning provide a new class of movement primitives. *Frontiers in computational neuroscience*, 6 (January):97, January 2012. ISSN 1662-5188. doi: 10.3389/fncom.2012.00097. URL <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3534186&tool=pmcentrez&rendertype=abstract>.

- [52] Stefan Schaal. A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010. ISSN 15324435. URL <http://dl.acm.org/citation.cfm?id=1953033>.
- [53] SeaGate. AIBO ERS-7 - 3D Warehouse. URL <https://3dwarehouse.sketchup.com/model.html?id=2bd7ed5ef2d55ced9461312eb423574c>.
- [54] R Shadmehr and S. P. Wise. Minimum Jerk Trajectory. In *Computational Neurobiology of Reaching and Pointing*. 2004. URL [http://www.shadmehrlab.org/book/minimum\\_jerk/minimumjerk.htm](http://www.shadmehrlab.org/book/minimum_jerk/minimumjerk.htm).
- [55] Freek Stulp and Evangelos Theodorou. Learning motion primitive goals for robust manipulation. ... *Robots and Systems* ..., 2(Section IV):1–7, 2011. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=6094877](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6094877).
- [56] Ioan Sucan. urdf - ROS Wiki. URL <http://wiki.ros.org/urdf/>.
- [57] R S Sutton and A G Barto. Reinforcement learning: an introduction. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 9: 1054, 1998. ISSN 1045-9227. doi: 10.1109/TNN.1998.712192.
- [58] RA A Téllez, Cecilio Angulo, and DE E Pardo. Highly modular architecture for the general control of autonomous robots. *Computational Intelligence and ...*, (Figure 1): 709–716, 2005. URL [http://link.springer.com/chapter/10.1007/11494669\\_87](http://link.springer.com/chapter/10.1007/11494669_87).
- [59] Ricardo Tellez. Aibo Quickstart Manual. Technical report, Grup de Recerca en Enginyeria del Coneixement (GREC), 2004. URL <http://www.ouroboros.org/AIBO-quickstart.pdf>.
- [60] F. Torrent-Fontbona, V. Muñoz, and B. López. Solving large immobile location-Allocation by affinity propagation and simulated annealing. Application to select which sporting event to watch. *Expert Systems with Applications*, 40:4593–4599, 2013. ISSN 09574174. doi: 10.1016/j.eswa.2013.01.065.
- [61] M Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International Journal of Humanoid ...*, 1(1):157–173, 2004. URL <http://www.worldscientific.com/doi/abs/10.1142/S0219843604000083>.
- [62] Reza Yazdani, Vahid Johari Majd, and Reza Oftadeh. Dynamically stable trajectory planning for a quadruped robot. *20th Iranian Conference on Electrical Engineering (ICEE2012)*, pages 845–850, May 2012. doi: 10.1109/IranianCEE.2012.6292471. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6292471>.



# Annexos

## Apèndix A

# Instal·lació de *ROS*, llibreries d'Urbi i paquet *aibo\_server*

Tot seguit es mostren els passos a seguir per tal d'instal·lar *ROS* i com afegir una carpeta al la variable `ROS_PACKAGE_PATH`<sup>1</sup>.

1. Preparar per instal·lar *ROS* (en aquest cas per Ubuntu 12.04 32 bits).

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main"
> /etc/apt/sources.list.d/ros-latest.list'
```

2. Configurar claus de *ROS*.

```
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

3. Instal·lar *ROS fuerte*.

```
sudo apt-get update
sudo apt-get install ros-fuerte-desktop-full
```

4. Per tal que els `packages` en la carpeta del propi *ROS* puguin ser trobats més còmodament. La comanda final és per poder seguir treballant en el mateix terminal.

```
echo "source /opt/ros/fuerte/setup.bash">>> ~/.bashrc
source ~/.bashrc
```

5. Es necessiten instal·lar alguns paquets per poder continuar, com és el `rosws`, que es part del paquet `rosinstall`.

```
sudo apt-get install python-rosinstall python-rosdep
```

---

<sup>1</sup>Aquí es troben les direccions de les carpetes on *ROS* cerca els `packages`.

6. Es crea una carpeta de treball, extensió de la pròpia de *ROS*.

```
rosws init ~/fuerte /opt/ros/fuerte
mkdir ~/fuerte/sandbox
rosws set  /fuerte/sandbox
```

7. Per acabar la instal·lació de *ROS*, es repeteix l'acció (4), però en aquest cas, per poder ser trobats els *packages* de la carpeta que s'ha creat.

```
echo "source ~/fuerte/setup.bash">>> ~/.bashrc
source ~/.bashrc
```

8. Tot seguit, s'ha d'instal·lar la llibreria d'urbi. En aquest cas, tan sols s'ha descarregar l'arxiu comprimit<sup>2</sup> i extreure'l en \.

9. Finalment per instal·lar el paquet *aibo\_server*, s'ha de copiar la carpeta en alguna de les direccions de *ROS\_PACKAGE\_PATH* i compilar seguint aquest procediment:

```
roscd aibo_server/
rosmake --pre-clean
```

---

<sup>2</sup><http://www.gostai.com/downloads/urbi/1.5/urbi-sdk-1.5-10258c7a-i486-linux-gnu-gcc-4.1.tar.gz>

## Apèndix B

## Bibliografia