

Aibo

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

8 de juny de 2014



Resum

L'estabilitat del robots...

Resumen

La estabilidad de los robots...

Abstract

The estability of the robots...

Índex

Resum	1
Resumen	1
Abstract	1
Prefaci	7
Motivació	7
Requeriments previs	7
Introducció	9
Estat de l'art	9
Robots	9
Modelatge de robots	11
Aprenentatge supervisat (<i>SL</i>)	14
Aprenentatge per reforç (<i>RL</i>)	15
DMP (<i>Dynamic Moviment Primitive</i>)	18
Path Integral Policy Improvement (<i>PI</i> ²)	21
Algorismes avançats	21
Objectius	22
Abast del treball	22
Estructura del treball	22
1 Estudis preliminars	23
1.1 AIBO	23
1.1.1 <i>Hardware</i>	24
1.1.2 <i>Software</i>	26
1.2 ROS	26
1.2.1 Estructura de ROS	27
1.2.2 Objectius de ROS	28
1.2.3 Eines de ROS	29
1.3 Estabilitat	29
1.4 Algorismes de resposta davant pertorbacions	29
1.4.1 Model del robot	30
1.4.2 Aprenentatge supervisat	30
1.4.3 Aprenentatge per reforç	30
1.4.4 DMP	30
1.4.5 Affordance	30

1.4.6	Elecció final	30
1.5	Codis amb DMPs implementades	30
1.5.1	DMPs de Scott Niekum	30
1.5.2	Package complet del robot PR2 del USC-CLMC	30
2	Disseny inicial	31
2.1	Model de l'Aibo	31
2.2	Acceleròmetre	31
3	Disseny final	32
3.1	Execució DMPs sense PI^2	32
3.2	Execució DMPs amb PI^2	32
3.3	Plataforma	32
4	Conclusions	33
	Agraïments	34
	Referències	35
	Annexos	40
A	Instal·lació de ROS, llibreries d'Urbi i paquet aibo server	40
B	Bibliografia	41

Índex de figures

1	REEM-C	10
2	BigDog	10
3	LittleDog	11
4	Esquema de cada arquitectura d'aprenentatge [29]	13
5	Gràfics explicatius de la funció no lineal $f(s)$	19
6	Efecte dels pesos w_i (representats per θ) i de l'objectiu (<i>goal</i>) en la tra- jectòria [39]	20
7	AIBO ERS-7 [32]	25

Índex de taules

1	Relació tipus de model amb arquitectura d'aprenentatge [29]	13
2	Rangs de funcionament de les articulacions [10]	26

Prefaci

Motivació

Moltes persones que tenen devoció pel món de la robòtica els hi ha vingut des de ben joves, aquest no és el meu cas. A mi, el que m'agradava era la informàtica. Sent petit vaig aprendre de forma autodidacte 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 en Projecte II, vam controlar un mecanisme a través d'un microcontrolador *Arduino*¹, amb programació basada en *Wiring* [4], llavors 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 d'escollir aquest TFG. 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* del *Boston Dynamics* o el *NAO* de *Aldebaran Robotics*², 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 el dia de demà. Aquest fet es veu tant en l'elecció dels llenguatges 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 i la millora d'un mateix per tal de poder arribar a un bon futur professional en aquest camp.

Requeriments previs

En el TFG aquí present, tot i que s'ha intentat donar una explicació a cada concepte que podria no haver-se entès, es necessari tenir uns certs coneixements previs. Aquests serien nocions en ROS³, Control Automàtic, llenguatge de programació Python, 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 DMPs. Alguns dels recomanats es poden trobar en les Referències o en la Bibliografia.

¹*Arduino* és la marca d'una família de microcontrolador

²En l'estat de l'art es dona una breu explicació d'aquests i altres robots

³*Robot Operative System*, s'explica detalladament en l'apartat 1.2

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.

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 [13]. Actualment, els robots s'utilitzen principalment per la realització d'accions de forma més exacte i barata o per treballs perillosos o repetitius. Ara bé, també existeix el cas de l'AIBO, entre d'altres robots, que pot ser utilitzat tant per entreteniment de l'usuari com 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.

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*), modelat de robots, aprenentatge supervisat, per reforç o *DMP*, generador de patrons centrals (*CPG*), algorismes genètics (*GA*)⁴, 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) Robots

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

⁴La majoria d'aquests conceptes seran explicats al llarg d'aquest apartat.

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 [28] sobre l'estabilitat d'un robot bípede a l'hora de caminar, corre i saltar. Basat en la teoria del *ZMP*.

REEM-C

Aquest humanoide és el creat per PAL Robotics [35]. 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, amb 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 [36].



Figura 1: REEM-C

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 [12]. 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 [5] i sobre la interacció amb els humans [27].

BigDog

És un dels grans robots que s'han creat al *Boston Dynamics*, prenent el que va ser inicialment desenvolupat en la DARPA [?]. Aquest "gos" va ser dissenyat per ús militar, en concret, per acompanyar als soldats portant la carga necessària en terrenys on no podria desplaçar-se un vehicle convencional. Aquest quadrúpede és dinàmicament estable^a gracies 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*^b), en concret DMP (*Dynamic Movement Primitives*)^c [?].

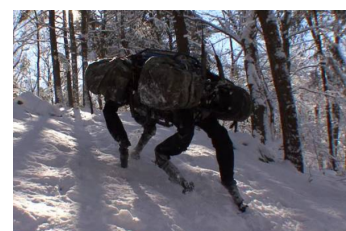


Figura 2: BigDog

^aSistema que és estable tenint en compte els efectes inercials i altres components dinàmiques que apareixen en el propi sistema [30].

^bAprenentatge per reforç s'explica en detall en l'estat de l'art. En molts casos es abreviat com *RL*.

^cDMP (*Dynamic Movement Primitives*) s'explica en detall en l'estat de l'art a la pàgina 18

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 [16].



Figura 3: 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 [25] se li aportant 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 [22] i [23] es centra en l'adaptabilitat del quadrúpede a diferents terrenys; (4) el MERO, estudiat en [14] per fer un 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 [21], com en la construcció des de zero [24], entre d'altres.

(2) Modelatge 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 correcta. 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.

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 en la figura 4.

Ara bé, existeixen tant diferents tipus de models, com també formes diferents de crear-los segons [29]:

- 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 el 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 en la taula següent:

- Arquitectura d'aprenentatge:

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

Modelatge indirecte Una de les tècniques per dur a terme modelatge 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.

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 1: Relació tipus de model amb arquitectura d'aprenentatge [29]

Aprenentatge amb professor distal La idea és crear un model invers, però guiat amb un model directe, per tal de minimitzar la manca d'unicitat del model invers.

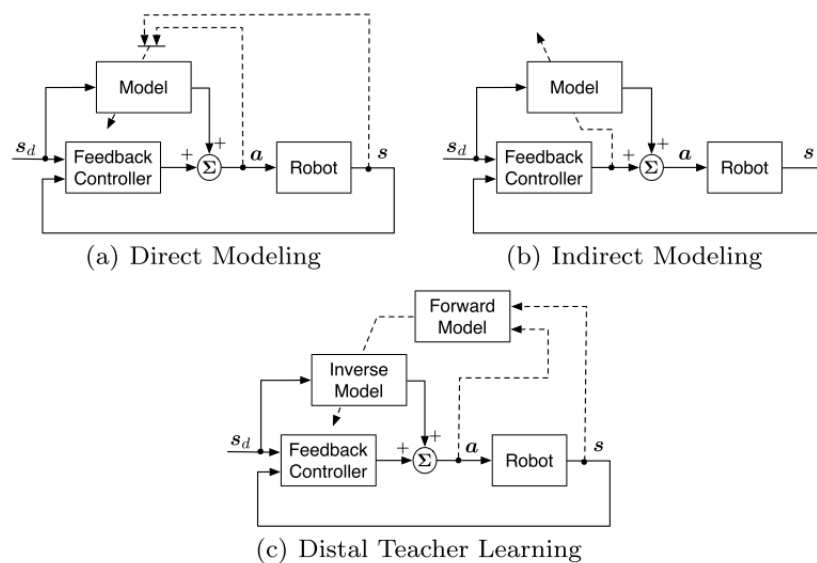


Figura 4: Esquema de cada arquitectura d'aprenentatge [29]

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 [11].

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:

WebotsTM [26] Simulador de robots mòbils desenvolupat per Cyberbotics Ltd. La física està basada en Open Dinamic 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 MindstormsTM o Pioneer2. Però té la desventatge que és un simulador de pagament.

SimRobot [19] Aquest és un simulador genèric de robots en 3D. Com el WebotsTM, 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 [17] És un simulador multi-robots en 3D. Aquest, al igual que WebotsTM, permet el modelatge del teu propi robot, tot i ser en llenguatge C, també es diferencia amb Gazebo pels models que inclou, que són el Pioneer2DX i el SegwayRMP.

Com s'ha mencionat, en les descripcions anteriors, WebotsTM inclou un model del Sony Aibo, dissenyat en [11]. Aquest té implementat l'estructura cinemàtica, propietats dinàmiques⁵, el seu control i l'aspecte gràfic. Per altra banda, també es pot 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.

(3) 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 [18]. 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 [6], 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 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].

⁵Masses i moments d'inèrcia

1. El model es representa com $g(x|\theta)$ ⁶, on $g(\cdot)$ és la classe d'hipòtesi i els valors de θ 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

$$E(\theta|X) = \sum_t L(r^t, g(x^t|\theta)) \quad (1)$$

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

$$\theta^* = \arg \min_{\theta} E(\theta|X) \quad (2)$$

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 simulada⁷.

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 [20]. 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 (2)

$$i^* = \arg \min_{i \in \{1 \dots n\}} D(x_t, x_i) \quad (3)$$

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 correcta o incorrecte.

(4) 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

⁶La x són les entrades, mentre θ són els paràmetres

⁷A 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[42]

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 [18]. 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 [15].

Aquest agent i el seu entorn poden ser modelat com un estat $s \in S^8$ i una acció $a \in A^9$. 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 política¹⁰ π que maximitza la recompensa acumulada escollint unes accions a en determinats estats s .

La idea clàssica d'aprenentatge per reforç es prenia des del punt de vista que l'agent consistia en un procés de decisions de Markov (*Markov Decision Process* o *MDP*)¹¹ 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 [40].

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 [18]. 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.

$$J = E \left\{ \sum_{h=0}^H R_h \right\}. \quad (4)$$

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

Aquest és introduït manualment i determina en quina proporció afecta el futur¹².

$$J = E \left\{ \sum_{h=0}^{\infty} \gamma^h R_h \right\}. \quad (5)$$

⁸Un estat s conté la informació necessària per descriure la situació actual i futures.

⁹Un estat del sistema es controlat o carregat per una acció a .

¹⁰Per política s'entén com en [8] "Manera de conduir un afer."

¹¹Conjunt 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.

¹²Com 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 [18].

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.

$$J = \lim_{H \rightarrow \infty} E \left\{ \frac{1}{H} \sum_{h=0}^H R_h \right\}. \quad (6)$$

A l'hora d'estimar la política π 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 del model són anomenats **model-based**.
- Per altra banda, entre d'altres mètodes els més utilitzats són el **Monte Carlo**, **Mètodes de diferència temporal**, **SARSA**, **R-learning** i **Q-learning**, sent aquesta última la més extesa, gracies a la seva sencillesa.

El comportament après és totalment depenent 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 continues per tal de guiar l'aprenentatge, en lloc d'una recompensa binària segons si s'ha complert o no la tasca [?]. Molts cops el comportament no és l'esperat, tot i que, per la nostra forma de pensar sembles que la solució és obvia. Per això, en alguns casos, s'utilitza l'aprenentatge per reforç invers, que aconsegueix extreure la funció recompensa gracies a un seguit de demostracions, pot ser no sigui la verdadera recompensa, però provoca l'actuació de la forma desitjada [18].

Alguns dels molts problemes que comporta el fet d'aplicar-se en robots [18]. 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 jeràrquica¹³ o amb tasques progressives¹⁴.

Per altra banda, dur 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

¹³S'assumeix que una part és fixa, mentre les altres s'aprenen, per tenir un solució inicial per després fer l'aprenentatge global.

¹⁴Alguns cops és més senzill aprendre una tasca complicada si es fan anteriorment algunes de no tant complicades.

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.

(5) 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 és 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.

El sistema dinàmic és pot interpretar com un PD¹⁵, 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 τ és el període del moviment i g és el paràmetre d'atracció del sistema.

¹⁵Controlador proporcional i derivatiu.

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \quad (7)$$

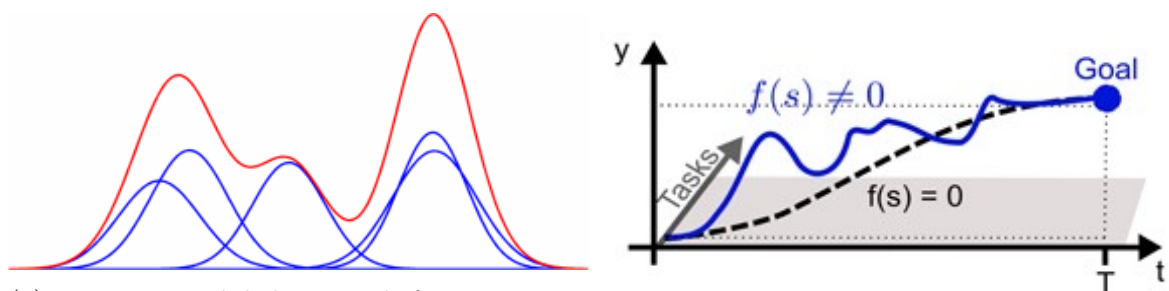
$$\tau \dot{x} = v \quad (8)$$

Si s'estudia el sistema dinàmic unidimensional de les equacions (7) (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 no depèn del temps, sinó de la variable de fase $s \in [0, 1]$ que representa la durada del moviment en tant per un definida per τ i per α ¹⁶ com es veu en (10), equació diferencial conegut com a sistema canònic (*canonical system*). A més, aquesta funció $f(s)$ pot aprendre per tal de dur a terme moviments complexos de forma arbitrària, ja que els pesos w_i es poden ajustar.

$$f(s) = \frac{\sum_i w_i \psi_i(s)s}{\sum_i \psi_i(s)} \quad (9)$$

$$\tau \dot{s} = -\alpha s \quad (10)$$

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 5a. 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 5b, per així, poder-se adaptar a noves situacions (com evadir obstacles, canvi d'objectiu, etc.).



(a) Representació de la suma de funcions gaussianes, on cada linea blava és una gaussiana i la vermella és la suma del conjunt [38]. (b) Representació de la DMP amb funció no lineal i sense [37].

Figura 5: Gràfics explicatius de la funció no lineal $f(s)$

¹⁶La α és una constant pre-definida

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 diferents 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 PI², que s'explica en el següent punt d'aquesta secció.

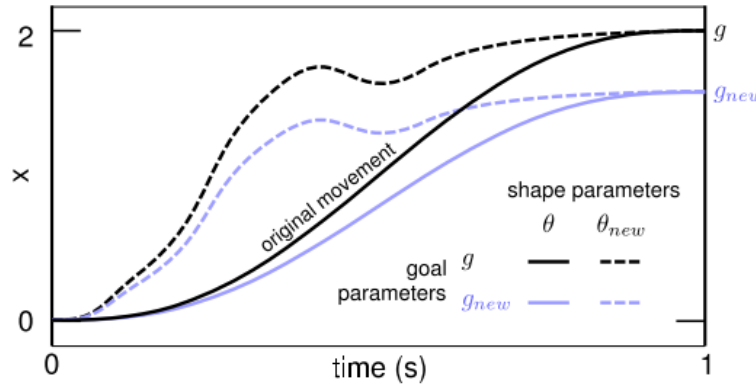


Figura 6: Efecte dels pesos w_i (representats per θ) i de l'objectiu (*goal*) en la trajectòria [39]

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

- 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 limits 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 [?] es proposa una modificació de l'algorisme original. L'única equació que es veu retocada és (7) que és substituïda per (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.

$$\tau \dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) \quad (11)$$

(6) Path Integral Policy Improvement (PI²)

L'algorisme PI² [39] té com a principal objectiu polir el pesos w_i (al llarg d'aquest apartat s'hi refereix com paràmetres θ_t), per tal que es minimitzi la funció de cost (12) de la trajectòria τ_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.

$$J(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} (r_t + \frac{1}{2}\theta_t^T R \theta_t) dt \quad (12)$$

La funció $J(\tau_i)$ és creada per l'usuari, segons la tasca que es vulgui desenvolupar, sent ϕ_{t_N} el cost final, r_t el cost immediat i $\frac{1}{2}\theta_t^T R \theta_t$ cost de control immediat¹⁷. Aquests dos últims, determinen principalment els valors que prenen els diferents θ_t durant la trajectòria, mentre el cost final ϕ_{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 concret¹⁸, els costos es presentarien d'una forma similar a (??). L'altre plantejament seria com un problema de RL, per exemple¹⁹, si es vol acabar en un punt (*goal*), amb poca velocitat i durant el recorregut amb l'acceleració minimitzada seria de l'estil de (??).

$$r_{300ms} = 100000000(G - y_{t_{300ms}})^2 \quad \phi_{t_N} = 0 \quad (13)$$

$$r_t = 0.5\ddot{y}_t^2 + \frac{1}{2}10000(\theta_t^T \theta_t) \quad \phi_{t_N} = 10000(y_{t_N}^2 + (g - y_t)^2) \quad (14)$$

Els mètodes de millora de política, com és el PI², consisteixen en un procés iteratiu d'exploració i actualització dels paràmetres. En l'exploració es proven K DMPs diferenciades per prendre els θ^{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 θ^{new} que són els actuals.

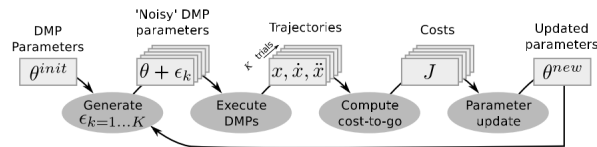


Figura 7: Algorisme genèric dels mètodes de millora de política [39]

¹⁷Regula el fet d'augmentar o disminuir els paràmetres θ_t , envers el benefici de fer-ho [?]]

¹⁸Extret de [?] apartat 5.2 *Learning Optimal Performance of a 1 DOF Via-Point Task*.

¹⁹Extret de [?] apartat 5.1 *Learning Optimal Performance of a 1 DOF Reaching Task*.

Per últim, en el cas concret del PI^2 , l'algorisme segueix la següent estructura, segons [39] 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 $\theta^{new} = \theta^{init} + \delta\theta$.

(7) 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 [41]:

"...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) extretes 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.

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 marcat 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)[33], però s'ha de millorar per fer-la més robusta.

,

Abast del treball

Estructura del treball

1 Estudis preliminars

1.1 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* el 1999, a partir d'aquest, i després de tres generacions, el 2003 s'arribà al *ERS-7*, molt més sofisticat que els predecessors, tot i que en el 2006 s'aturà la producció de la família AIBO. El *ERS-7* és el model que s'estudia i s'utilitza en el present treball.

Aquest es 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 solen ser dissenyats per desenvolupar tasques de seguretat o treballs perillosos, ara bé, en aquests casos no es pot tolerar cap tipus d'error en les operacions crítiques. Mentre els que estan dissenyats per usos d'entreteniment, en el cas que es produís algun error no seria un amenaça per la vida [9].

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

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

1.1.1 *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
- 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: 208×160 , 104×80 , 52×40
 - 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)

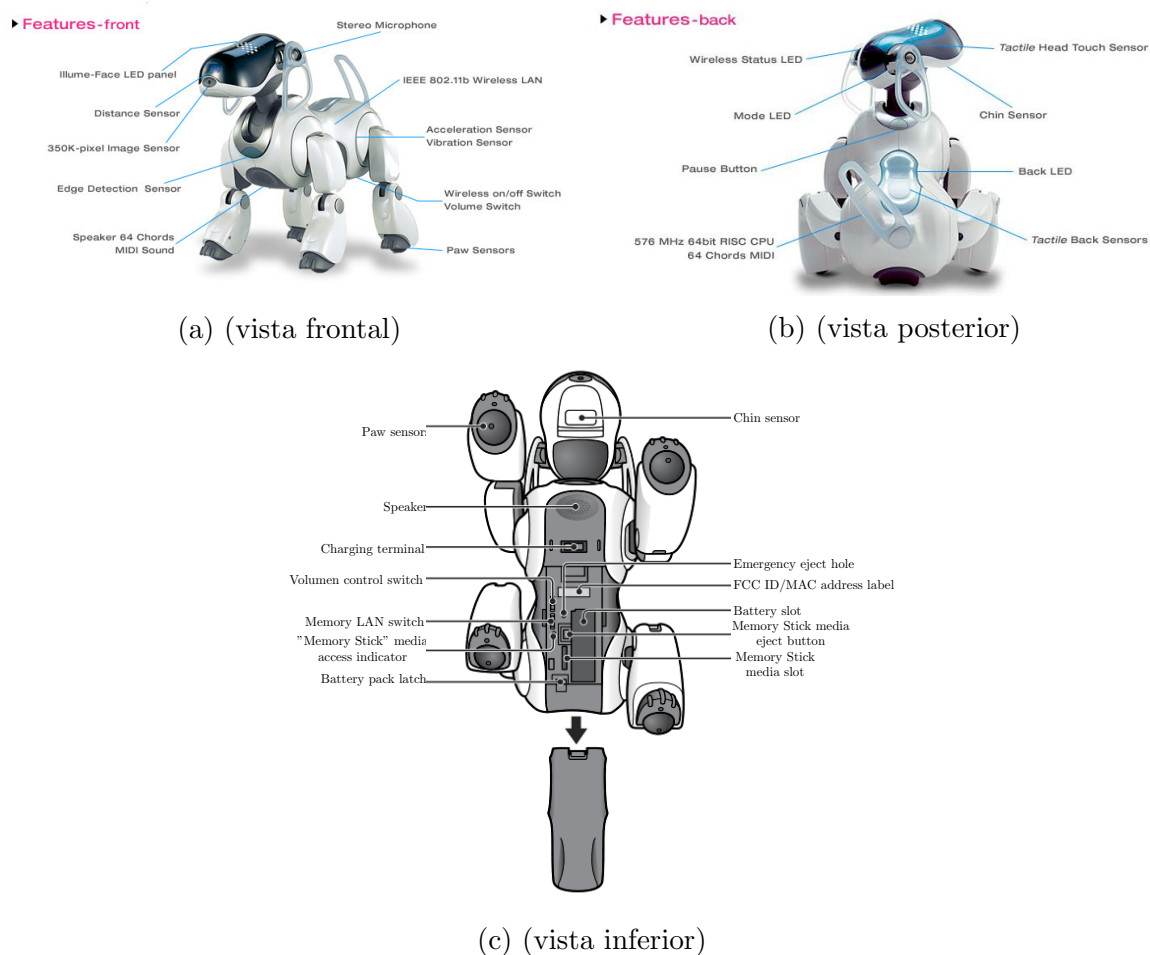


Figura 8: AIBO ERS-7 [32]

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:

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 2: Rangs de funcionament de les articulacions [10]

1.1.2 Software

1.2 ROS

Robot Operating System [34] 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 STanford Artificial Intelligence Robot (*STAI*) o Personal Robotics (*PR*). Més endavant, *Willow Garage*, una empresa inversora en robòtica, va proveir recursos 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.

1.2.1 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**. Aquests són simplement estructures de dades, que poden anar des d'**integrer**, **floats**, **booleans** fins a **arrays**.
- **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*; 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 altre 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.

1.2.2 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** i ser compartits de forma senzilla.

D'altra banda, també es tenen unes altres finalitats [31]: (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 heterogènia²⁰.

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 vers, 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

²⁰Una xarxa heterogènia és una xarxa de connexió d'ordinadors i altres dispositius amb diferents sistemes operatius i/o protocols.[7]

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.

1.2.3 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 varies 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 és té, juntament amb altres dades 3D que se li aportin, per poder visualitzar el conjunt.

- **roscap i rxscap**

Roscap és la comanda que et permet emmagatzemar i reproduir de nou les dades d'un **message** en un arxiu **bag**. Per altra banda, rxscap 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.

1.3 Estabilitat

1.4 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.

1.4.1 Model del robot

1.4.2 Aprenentatge supervisat

1.4.3 Aprenentatge per reforç

1.4.4 DMP

1.4.5 Affordance

1.4.6 Elecció final

1.5 Codis amb DMPs implementades

1.5.1 DMPs de Scott Niekum

1.5.2 Package complert del robot PR2 del USC-CLMC

2 Disseny inicial

2.1 Model de l'Aibo

2.2 Acceleròmetre

3 Disseny final

3.1 Execució DMPs sense PI^2

3.2 Execució DMPs amb PI^2

3.3 Plataforma

4 Conclusions

Agraïments

Referències

- [1] J.C. Albiez, T. Luksch, K. Berns, and R. Dillmann. Reactive reflex-based control for a four-legged walking machine. *Robotics and Autonomous Systems*, 44(3-4):181–189, September 2003. ISSN 09218890. doi: 10.1016/S0921-8890(03)00068-X. URL <http://linkinghub.elsevier.com/retrieve/pii/S092188900300068X>.
- [2] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, volume 5. 2004. ISBN 0262012111. doi: 10.1007/s10994-009-5137-3. URL <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0262012111>.
- [3] Muh. Anshar and Mary-Anne Williams. Extended Evolutionary Fast Learn-to-Walk Approach for Four-Legged Robots. *Journal of Bionic Engineering*, 4(4):255–263, December 2007. ISSN 16726529. doi: 10.1016/S1672-6529(07)60039-0. URL <http://linkinghub.elsevier.com/retrieve/pii/S1672652907600390>.
- [4] Arduino. Arduino - HomePage. URL <http://www.arduino.cc/>.
- [5] J. Chestnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep Planning for the Honda ASIMO Humanoid. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.
- [6] Matthieu Cord and Pádraig Cunningham. *Machine learning techniques for multimedia*. 2008. URL <ftp://icksie.no-ip.org/EBooks/Computers/ArtificialIntelligence/Semi-supervisedlearning.pdf>ftp://icksie.no-ip.org/EBooks/Computers/ArtificialIntelligence/Cord_Cunningham-Machine_Learning_Techniques_for_Multimedia-9783540751700.pdf.
- [7] 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.
- [8] Institut d’Estudis Catalans. Diccionari de la llengua catalana. URL <http://dlc.iec.cat/index.html>.
- [9] 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.
- [10] Gostai. URBI Doc for Aibo ERS2xx ERS7 and URBI 1.0. URL <http://www.gostai.com/doc/en/aibo/>.

- [11] 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>.
- [12] HONDA. History of ASIMO Robotics — ASIMO Innovations by Honda. URL <http://asimo.honda.com/asimo-history/>.
- [13] International Organization for Standardization. ISO 8373:2012: Robots and robotic devices — Vocabulary. Technical report, ISO, Genève, 2012.
- [14] 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>.
- [15] 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>.
- [16] 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>.
- [17] Oussama Khatib, Oliver Brock, Kyong-Sok Chang, Francois Conti, Diego Ruspini, and Luis Sentis. Robotics and interactive simulation, 2002. ISSN 00010782.
- [18] 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.
- [19] 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.
- [20] 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>.
- [21] 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.

- [22] 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>.
- [23] 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>.
- [24] 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>.
- [25] 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>.
- [26] Olivier Michel. Cyberbotics Ltd. Webots: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems*, 1:40–43, 2004.
- [27] 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>.
- [28] 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.
- [29] 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>.
- [30] A. Purushotham and G. Venkata Rao. Dynamic stability analysis of a quadruped robotic manipulator system: analytical approach. *International Journal of Applied Engineering Research*, 2009.

- [31] 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.
- [32] Rainersen. Imatges Aibo ERS-7. URL <http://rainersen.de/aibo/>.
- [33] Carlos Mario Ramos Olave. Diseño, implementación y control visual de posición de un sistema placa-bola. 2013.
- [34] Robot Operating System. ROS.org. URL <http://www.ros.org/core-components/>.
- [35] PAL Robotics. REEM-C, . URL <http://pal-robotics.com/en/robots/reem-c>.
- [36] PAL Robotics. REEM-C — PAL Robotics Blog, . URL <http://blog.pal-robotics.com/blog/2013/11/21/unplug-play-reem-c/>.
- [37] 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>.
- [38] 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>.
- [39] 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.
- [40] 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.
- [41] RA Téllez, Cecilio Angulo, and DE Pardo. Highly modular architecture for the general control of autonomous robots. *Computational Intelligence and ...*, pages 709–716, 2005. URL http://link.springer.com/chapter/10.1007/11494669_87.
- [42] 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.

Annexos

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`²¹.

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 `roinstall`.

```
sudo apt-get install python-roinstall python-rosdep
```
6. Es crea una carpeta de treball, extensió de la propia 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
```

²¹Aquí es troben les direccions de les carpetes on ROS cerca els `packages`.

8. Tot seguit, s'ha d'instal·lar la llibreria d'urbi. En aquest cas, tan sols s'ha descarregar l'arxiu comprimit²² i extreure'l en \.
9. Finalment per instal·lar el paquet d'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
```

B Bibliografia

²²<http://www.gostai.com/downloads/urbi/1.5/urbi-sdk-1.5-10258c7a-i486-linux-gnu-gcc-4.1.tar.gz>