Technical Report

IRI-TR-10-13

Quadern de Treball CEABOT 2009

TWIKI

Jordi Pegueroles
Edgar Simó

Institut de Robòtica i Informàtica Industrial

Abstract

Resum de l'experiència en la participació del concurs CEABOT 2009.

Institut de Robòtica i Informàtica Industrial (IRI)
Consejo Superior de Investigaciones Científicas (CSIC)
Universitat Politècnica de Catalunya (UPC)
Llorens i Artigas 4-6, 08028, Barcelona, Spain
Tel (fax): +34 93 401 5750 (5751)
http://www.iri.upc.edu

Corresponding author:
E. Simó
tel: +34 93 401 xxxx
esimo@iri.upc.edu
http://www.iri.upc.edu/people/esimo

Contents

1 Introducció

1.1 Descripció del robot

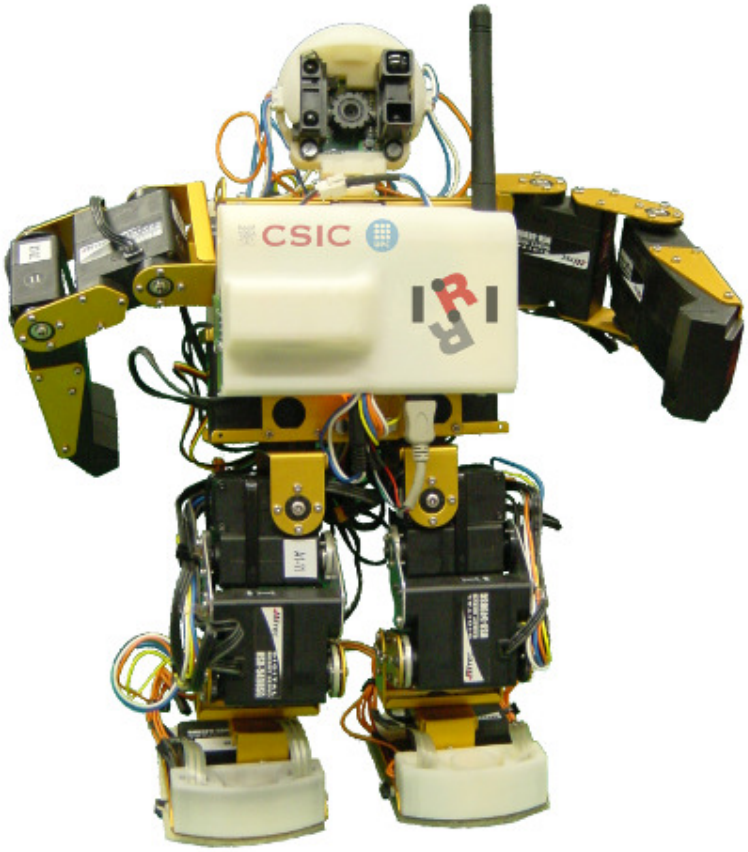

Figure 1: Twiki el robot

El robot esta basat en la plataforma robòtica Robonova de Hitech. Disposa de diverses millores, tant a nivell mecànic com a nivell electrònic. En la realització de les millores s'ha seguit el criteri de crear un robot el més antropomòrfic possible.

1.2 Millores mecàniques

Grau de llibertat extra a les cames: Aquest grau de llibertat de les cames permet realitzar girs de precisió de forma ràpida i elegant. Com a contrapartida s'ha augmentat el pes del robot, elevat el centre de gravetat i augmentat el consum energètic.

pan & tilt: S'han afegit 2 microservos acompanyats d'un element estructural que permet muntar un cap sensoritzat al robot. El cap disposa de dos graus de llibertat per realitzar moviments. Això permet orientar la càmera sense haver de moure tot el cos del robot, guanyat velocitat i precisió a l'hora de detectar els elements de l'entorn.

bateries: L'increment de pes a comportat una necessitat de major potència dels servos. Aquesta major potencia l'aconseguim instal·lant una nova cel·la a la bateria original del Robonova.

Per tal de distribuir millor el pes, s'ha creat una motxilla que permet repartir les sis cel·les totals per l'esquema.

motxilla: Per protegir el ordinador incrustat s'ha dissenyat una motxilla que el protegeix dels cops i possibles caigudes.

1.3 Millores electròniques

Ordinador incrustat: S'ha instal·lat un ordinador gumstix verdex per poder crear algorismes de molt alt nivell. Aquest ordinador ens permet també interactuar de forma senzilla i eficaç amb la càmera usada per als algorismes de visó.

Càmera: El robot disposa d'una càmera USB de Logitech per a poder realitzar algorismes basats en visió.

Sensors de pressió als peus: Cada peu disposa de 6 sensors de pressió, distribuïts 4 a la planta i 2 més al frontal del peu.

Sensors de distància: En el cap s'han instal·lat 2 sensors IR de mig i llarg alcans.

Sensors d'acceleració: El robot disposa de 2 sensors d'inclinació que permeten saber quan i com ha caigut el robot.

2 Mobilitat

Se trata de una prueba de movilidad, con dos niveles. En el primer nivel, el robot realizará una carrera ida y vuelta salvando unos obstáculos colocados de forma aleatoria. En el segundo nivel, el robot realizará una carrera (solo ida), teniendo que superar una escalera con peldaños de 3 cm. de altura.

2.1 Primer nivell: Obstacles

2.1.1 Descripció de la prova

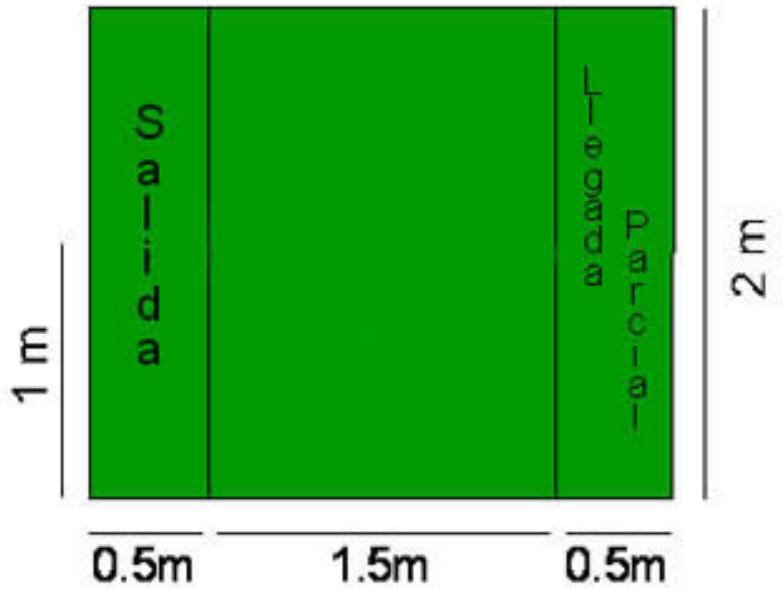

Figure 2: Disposició de la prova de cursa

Aquesta prova consisteix en fer un recorregut de 1.5 metres d'anada i de tornada en una pista verda com la de la figura 2. L'any 2009 es va introduir el concepte de obstacles a la prova, anteriorment només calia anar i tornar sense desviació. Els obstacles obliguen a canviar l'estratègia completament. Quan abans només calia optimitzar la velocitat i minimitzar la desviació, ara cal intentar trobar un camí òptim.

Els obstacles són caixes blanques de color blanc de mides 25x25x50 cm. N'hi han 6 i només es poden posar a sobre una matriu de 25x25 cm. Per tant sempre hi ha com a mínim dos camins rectes per evitar els obstacles.

2.1.2 Plantejament

La prova es va plantejar de intentar imitar el comportament humà en una situació com aquesta, és a dir, intentar trobar el camí abans de començar a moure. Hi havien dues possibilitats per trobar les caixes: fer servir els sensors infraroig o fer servir la càmera. El sensor infraroig es va descartar per la complexitat de càlcul que comporta. La càmera inicialment va sembla idoni ja que no calia prendre massa mostres.

Els sensors infraroig s'utilitzarien per tal de detectar quan s'arribava al final i calia girar. Per no desviar-se es va optar per fer servir la càmera per detectar la inclinació de la línia de la zona d'arribada parcial. Es va descartar fer servir altres sensors.

2.1.3 Processament d'imatge

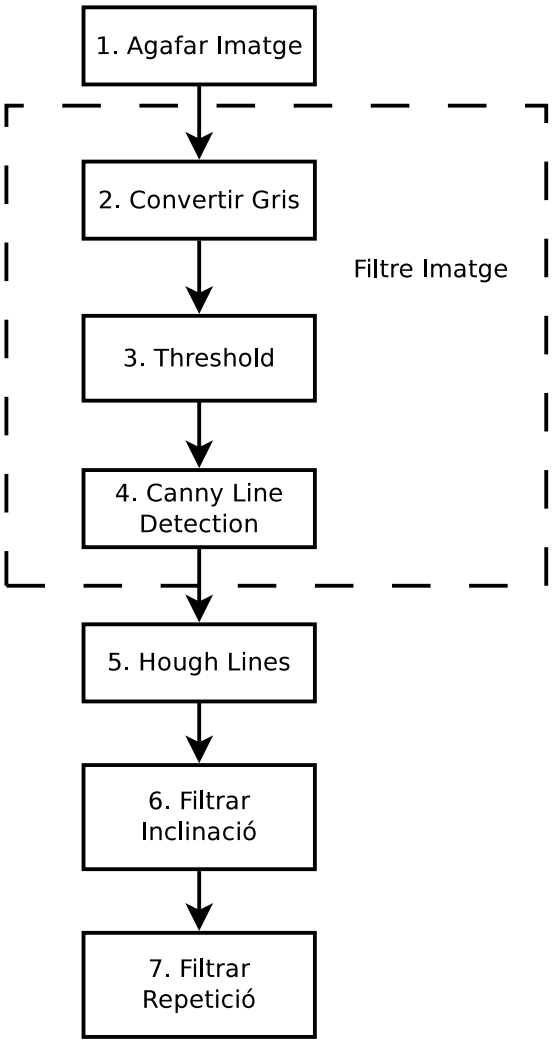

Figure 3: Obtenció de les línies de la imatge

La clau de aquesta prova era trobar ràpidament el camí mínim per tal de començar a caminar. Per fer la visió es va decidir en fer servir OpenCV ja que és una llibreria coneguda, madura i versàtil. El procés emprat es pot veure a la figura 3. A continuació s'explica cada etapa en detall:

1. Agafar Imatge: En aquesta etapa agafem la imatge. Degut a l'ample de banda del USB 1.1 que té el Gumstix (PXA270), només podem fer servir una resolució de 320x240 BGR ja que la càmera no permet obtenir imatges directament en gris.

2. Convertir Gris: Degut al fet que no podem tenir les imatges directament en gris i que tampoc ens cal d'informació en color convertir l'imatge de BGR a gris.

3. Threshold: Per simplificar els calculs i eliminar les parts mes fosques i clares apliquem un threshold.

4. Canny Line Detection: Originalment es feia servir el Sobel que és un filtre més ràpid però al voler obtenir millor precisió es va decidir emprar el filtre Canny. Aquests filtres serveixen per detectar els contorns de les caixes a les imatges.

5. Hough Lines: Aquesta etapa permet crear les línies que han estat detectades amb el filtre anterior.

6. Filtrar Inclinació: Es descarta informació no rellevant, en aquest cas línies amb inclinacions que no estan a prop dels valors que esperem.

7. Filtrar Repetició: Degut al fet que a vegades el filtre Hough trobava línies sobreposades es va fer un filtre per eliminar línies que es tallen o estan massa a prop l'una de l'altre.

Els resultats del la visió van ser prometedors. Al prioritzar precisió i tenir la il·luminació controlada els resultats eren molt fiables i per tant a l'algoritme principal es podia considerar els resultats de processament d'imatge gairebé com reals.

2.1.4 Algoritme

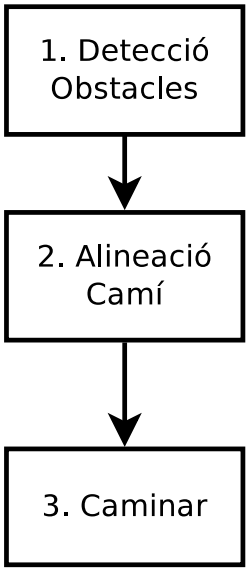

Figure 4: Algoritme principal de la prova de obstacles

L'algoritme en és bastant senzill, el problema cau en els detalls. Per superar els obstacles ens basem en el fet que sempre hi han d'haver dos camins rectes lliures. A la figura 4 es pot veure les tres etapes de l'algoritme:

1. 1. Defecció Obstacles En aquesta etapa emprem el pan and tilt per detectar els camins lliures.

2. 2. Alineació Camí Quan sabem en principi quin és el camí a seguir fem passos laterals i ens alineam amb el camí. Fem servir els sensor infraroig de llarga distància per comprovar que sigui el camí.

3. 3. Caminar Finalment s'intenta caminar en línia recta pel camí trobat, arribar al final, girar i tornar.

L'algoritme pot semblar senzill però amaga molta complexitat. Per detectar l'obstacle es fan 5 mostres a intervals regulars amb la càmera. En aquestes mostres es detecta les línies de la base de la caixa orientades cap a l'inici. Aquestes línies es mira si cauen en carrils que es projecten a sobre la imatge que corresponen a la matriu de la pista. Si una línia es detecta dins

d'un carril es considera que esta bloquejat. Així s'obté un mapa inicial de la pista. Es troba el camí lliure més proper al robot i es procedeix a la següent etapa.

Per alinear-se amb el camí es fan servir passos laterals i es fan mostres amb el sensor infraroig de llarga distancia. Es van fent passos fins que es troba l'escletxa i aleshores es passa al caminar. Els moviments laterals calen que no es desviïn gaire o poden complicar tot el procediment.

Finalment es camina. Per orientar-se es fa servir la línia de la zona d'arribada parcial. Observant la seva inclinació es pot veure si el robot va molt desviat per tal de corregir-ho. Es fan servir els sensors infraroigs quan s'agafa la imatge per detectar la línia d'arribada parcial, ja que cal que el robot s'estabilitzi. Una vegada que es detecta la paret final, el robot gira gairebé 180 graus i ara fa servir la línia d'arribada per corregir la seva inclinació. Continua caminant fins que arriba al final.

2.1.5 Resultats

Al laboratori es van obtenir resultats molt bo, però a l'hora de la competició es va trobar un molt important en tot el algoritme. Hi havien dos problemes, el primer és que la pista de sumo (rodona) i la de obstacles (quadrada) estaven sobreposades. Això volia dir que quan detectàvem la línia del fons per orientar-se es podia confondre amb la de la pista de sumo. Això principalment va sembla molt greu, però retocant alguns valors del processat d'imatge van poder minimitzar l'efecte especialment a la zona central de la pista. No teníem control sobre això, però al no ser-hi al reglament es pot considerar un problema de la organització. Es va parlar per esperar que a futures competicions no tornès a passar.

El segon problema va ser la manca de control de la il·luminació, això va comportar més problemes. Al no tenir la llum controlada, els resultats del processat d'imatge podien variar molt i calia recalibrar-lo a diferents moments. De la primera vegada que es va fer la prova a la segona va variar molt la lluminositat i gairebé ens va provocar la desqualificació. Resulta que els filtres que fèiem servir es feien molt lents si es trobaven moltes línies. El robot va trigar gairebé tot el temps processant imatges i per poc va arribar a la zona d'arribada a temps abans de que el desqualifiquessin. N'hi han diverses solucions per aquest problema. La primera seria intentar controlar la llum i l'entorn per tal de no tenir que retocar els paràmetres dels algoritmes. La segona solució seria treballar amb els algoritmes de visió per ser molt més robust en front de condicions canviants. En concret posar límits al temps de còmput.

Tot i els problemes detectats, el resultats van ser molt bons ja que l'algoritme era molt fiable. Es va acabar la prova dos cops sense penalitzacions que va donar molts punts.

2.2 Segon nivell: Pujar i baixar escales

2.2.1 Descripció de la prova de escales

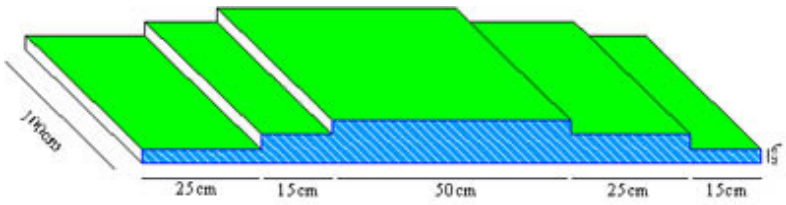

Figure 5: Disposició de la prova

A la figura 5 podem veure la topologia de la prova. Cal notar no que es tracta d'un escenari **no simètric**, fet important a l'hora de dissenyar l'algoritme.

2.2.2 Consideracions inicials

Com ja s'ha comentat anteriorment, la filosofia que hi ha darrera del desenvolupament del robot Twiki, es la de mantenir el robot el mes antropomòrfic possible. És a dir, ubicar els sensors de la forma més similar possible a la distribució dels òrgans perceptius humans.

2.2.3 Hora de fer Brainstorming

Un cop coneguda la prova, cal pensar una estratègia per tal de superar-la amb el mínim risc, màxima fiabilitat i la major velocitat possible. Busquem el millor algoritme (que se'ns acudeixi).

Opció A: posar sensors IR als peus per tal de llegir la distància fins al primer esglaó, pujar-lo i repetir per als següents esglaons. Per a trobar els esglaons de baixada, fer lectures de la distància del cap fins a terra, fins trobar una lectura sensiblement més elevada.

Aquesta opció es va descartar per no mantenir l'antropomorfisme i per ser una solució poc robusta.

Opció B: Usar els sensors instal·lats als peus per tal de detectar els esglaons.

A primera vista sembla una bona solució, ja que: mantenim l'antropomorfisme i es sabut que els sensors FSR, o més concretament els conversor AD dels FSR proporciona lectures ràpides, freqüents i altament fiables.

2.2.4 Elecció dels sensors a usar

Sensors usats en la prova:

- Sensors de pressió en la planta dels peus. (x4 FSR per peu)
- Sensors de pressió al frontal del peu. (x2 FSR per peu)
- Sensors d'infrarojos de curt alcans per a llegir distàncies.

La resta de sensors instal·lats en el robot no es creu necessari usar-los.

2.2.5 Detecció dels esglaons

El concepte dels FSR es molt bonic, però a l'hora de la veritat com detectem realment els esglaons? De sobte comencen a sorgir els dubtes, els nervis... Vam començar pel principi.

Quina sensibilitat tenen els sensors? Les primeres proves ens van ensenyar que calia un pes major de 2 quilogram per peu per tal de detectar alguna variació ens les lectures. Era evident que aplicar tot aquest pes en un peu, suposant que el robot pesés prou, provocaria problemes d'inestabilitat. Per tant vam haver de augmentar considerablement la sensibilitat. Sense entrar en els detalls tècnics, dir que calia variar el valor de les resistències divisores de tensió a l'entrada del ADC.

Primer vam començar detectant les parets verticals dels esglaons. Ara, amb la nova sensibilitat era possible detectar la paret del primer esglaó, amb 'una puntada de peu'. Es a dir, fent caminar el robot amb passos curts, es possible detectar quan colpeja el primer esglaó, i per tant saber quan hem de començar a pujar.

Ara per tant, calia detectar quan començar a baixar les escales. La primera idea va ser anar caminant fent passes curtes fins que algun dels sensors retornes una lectura corresponen a pressió nul·la. Això significaria que el peu estava a l'aire. El problema d'aquest mètode era que quan arribava el robot a detectar que tenia el peu a l'aire, era tant just, que sempre s'acabava desestabilitzant i caient. Calia un nou procediment. De sobte va sorgir la idea: detectar l'inici del l'esglaó de baixada amb la punta de la planta del peu. Aquest va resultar un procediment lent, però altament robust.

2.2.6 Implementació de l'algorisme

Ja tenim tots els elements per tal de composar el nostre algoritme. L'algoritme es va implementar usant una màquina d'estats.

```
tria estat:
        estat approach1:
            camina
            si (esglaó detectat)
                estat -> upstairs

        estat upstairs:
                puja esglaó
                si (esglaons pujats >= 3)
                        estat -> move_fwd1
                sino
                        estat -> approach1

        estat move_fwd1:
        /* esta a dalt de tot i sabem que sempre te la mateixa llargada */
                camina 4 passos llargs
                estat -> approach2

        estat approach2:
                camina
                toca amb la punta del peu
                llegeix FSR
                si (lectura > limit)
                        torna_peu_enrera
                        estat -> downstairs
                sino
                        posa peu a terra

        estat downstairs:
                baixa_esglaó
                si (pas = 2)
                        si (primer_pas_llarg)
                                estat -> approach3
                        sino
                                estat -> approach4
                si (pas = 1)
                        si (primer_pas_llarg)
                                estat -> approach4
                        sino
                                estat -> approach3

                si (ultim_esglaó)
                        estat -> move_fwd2

        estat approach3:
                pas_curt
```

```
        estat -> approach2

    estat approach4:
        pas_llarg
        estat -> approach2

    estat move_fwd2:
        fer 2 passos llargs
        estat -> final

    estat final:
        prova acabada
        moure braços fent moviment de victòria!
```

3 Sumo

El objetivo de esta prueba es que cada robot sea capaz de enfrentarse a otro dentro del Área de Combate para obtener puntos Yuhkoh. Los combates consistirán en 2 asaltos de 3 minutos cada uno con un tiempo máximo entre asalto y asalto de 1 minuto. Ganará el combate el robot con más puntos Yuhkoh en el total de los dos asaltos. En caso de empate a puntos, se realizará un asalto extra donde el ganador será el primero que consiga un punto Yuhkoh.

3.1 Algoritme

Per guanyar un combat sumo hi han dues opcions: empènyer el contrincant fora del ring o fer-ho caure a terra. Degut a les mides del ring generalment els combats sempre es guanyen quan un robot cau a terra, ja que no arriben mai a sortir del ring.

La opció de fer empènyer el contrincant fora del ring es va descartar per dues raons. Primer de tot, les mides del ring no fomentaven poder fer fora al contrincant. La mecànica del robot tampoc permet fàcilment moure un altre robot. Els servos no poden fer suficient força. Per tant la única opció vàlida era intentar fer caure al contrincant.

Per tal de tenir algoritmes senzills i fàcils de comprovar la seva validesa es va decidir de fer servir màquines d'estat finites. Aquests algoritmes es caracteritzen pel fet de que en qualsevol moment tenen un estat determinat. Les condicions de cada moment i l'estat actual determinen quin és el següent estat de l'algoritme. Són molt fàcils d'implementar amb codi i per tant permeten un desenvolupament més ràpid.

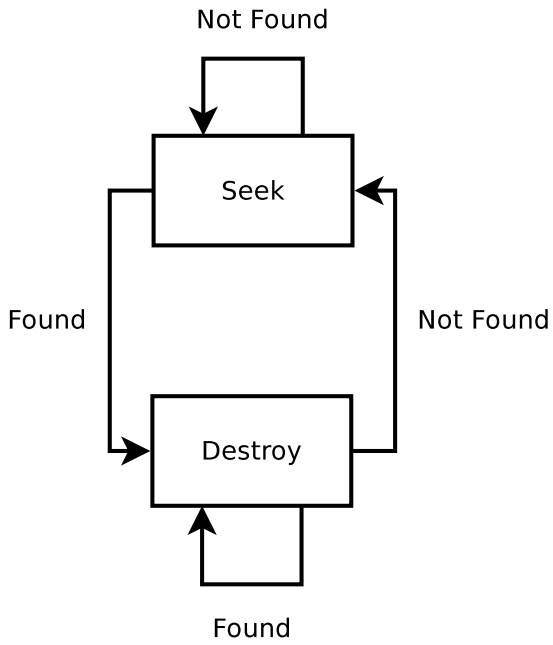

Figure 6: Algoritme sumo més senzill possible (seek and destroy)

La estratègia més senzilla possible era la del seek and destroy que es veu a la figura 6. Aquest algoritme consisteix en dos estats que alternen segons si el robot veu un objecte o no. A l'estat "seek" el robot intenta trobar el contrincant, quan el troba passar a l'estat "destroy" on intenta destruir el contrincant. Si el perd de vista de nou, torna a l'estat "seek".

Per tal de fer menys predicible el robot, es va decidir elaborar sobre el algoritme base "seek and destroy". Es va veure que si es feia servir el mateix tipus d'atac de llarga distancia quan el contrincant estava a prop, generalment el robot es tirava a si mateix al terra. Per tant es va

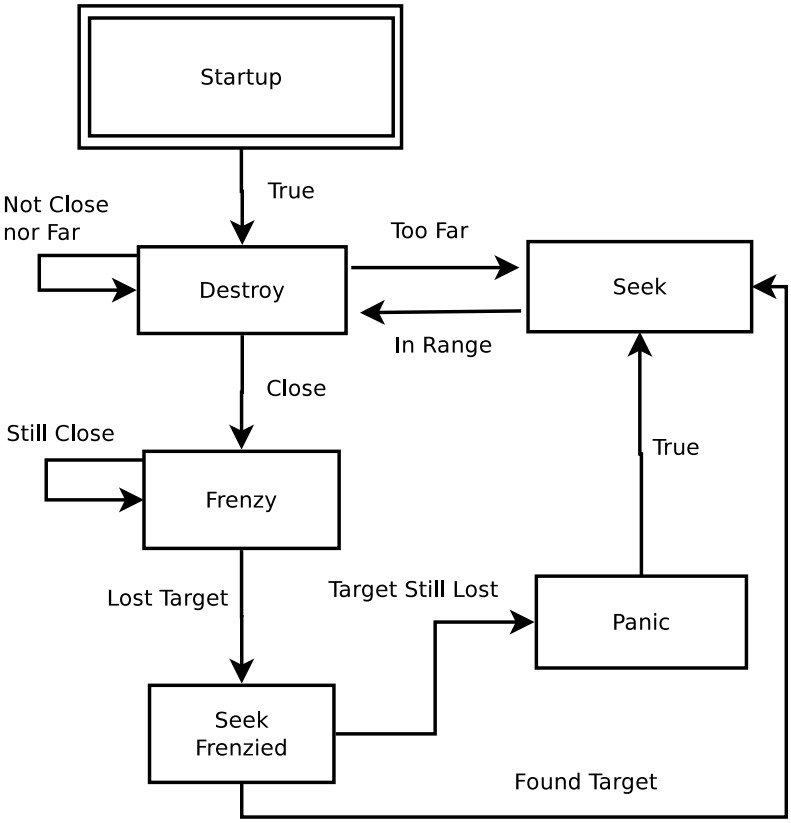

Figure 7: Algoritme sumo més finalment emprat

dividir el "destroy" en dos parts: "destroy" i "frenzy". El "frenzy" consisteix en intentar fer atacs curts i poderosos per a quan el contrincant esta a sobre.

Es va veure que quan els robots estaven molt a prop, a vegades es perdia de vista. Això feia que entres al "seek" on es posava en una posició molt vulnerable. Per evitar això es va afegir un estat de "seek frenzied" on es feia un "seek" però menys intens emprant un pose defensiu. Si tot i això no recuperava la posició del contrincant, es va afegir un estat "panic" on el robot intentava allunyar-se del contrincant per poder-ho detectar més fàcilment a distancia. Així s'evitava un estat de vulnerabilitat.

La màquina d'estat final es pot veure a la figura 7. Aquest algoritme és més complicat que el bàsic però conté millores substancials evitant moments de molta vulnerabilitat del robot. Aquest algoritme va funcionar molt bé a la competició demostrant la seva robustesa.

3.2 Moviments

Per tal de fer un bon combat sumo, el robot necessitava moviments només pel sumo. Aquest moviments havien de complir amb els següents requeriments:

1. Estabilitat: Tots els moviments han de ser molt estables ja que en qualsevol moment el robot pot rebre un cop del contrincant. Aquesta es la propietat més important de totes.

2. Velocitat: Els moviments han de ser molt ràpids per poder respondre a canvis en la pista de sumo.

3. Flexibles: Els moviments han de poder-se emprar per altres utilitats, per tal de poder optimitzar.

Els moviments i poses que es van fer servir al final van ser:

1. Standard Pose: Posició estàndard per poder moure fàcilment.
2. Forward: Moviment curt per anar endavant.
3. Right: Moviment curt per girar aproximadament 10-15 graus cap a la dreta.
4. Left: Moviment curt per girar aproximadament 10-15 graus cap a l'esquerra.
5. Defense: Posició ultra defensiva amb un centre de gravetat molt baix.
6. Attack3: L'atac estàndard que consisteix en donar un cop amb els dos punys.

Aquests moviments ens van donar problemes degut a la seva implementació. Els moviments s'havien de modificar fent servir el robobàsic per tal d'intentar optimitzar l'estabilitat. Degut al desgast del robot i altres condicions fora del nostre control, no van ser prou estables i van ser una de les principals causes del mal resultat en el combat sumo.

3.3 Detecció

Per detectar hi havia dues eines: la càmera i els sensors infraroigs. Tots estaven posats a sobre de un pan and tilt de tal manera que es podia orientar a l'espai per mirar cap a on calia.

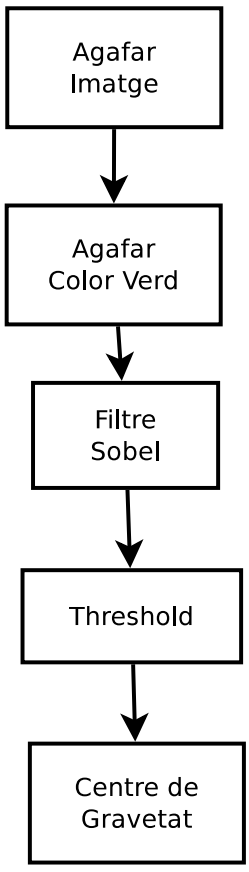

Figure 8: Algoritme de visió per sumo

L'algoritme per trobar el contrincant havia de ser molt senzill. Es va decidir de fer un escombrat horitzontal amb el pan per tal de intentar detectar el contrincant. Això es fa utilitzant

exclusivament la càmera. Una vegada que es trobava el contrincant es orientava el robot cap al contrincant i es feia un altre escombrat. Aquest algoritme es repetia fins tenir-ho centrat ja que el gir no era determinista i així es tancava un llaç de control amb la càmera.

Finalment quan es tenia centrat, es feia servir els sensors infraroigs per detectar la distancia i veure si calia apropa o es podia atacar directament. Si es perdia es tornava a fer els escombrats per trobar-ho de nou. D'aquesta manera es podia trobar el contrincant amb molta fiabilitat.

En quant als algoritmes de visió per computador, es va emprar algoritmes ràpids i aproximats per poder reaccionar amb poca latència a canvis. Es pot veure a la figura 8 els passos concrets. Cal destacar que es descarta els canals de blau i vermell per només emprar el canal verd. Això es degut al fet de que la pista de sumo és verd i els robots generalment són de altres colors molt diferents. Amb aquest pas separàvem la pista de qualsevol objecte diferent.

Posteriorment per segmentar els objectes de la imatge es fa ser un filtre Sobel que es caracteritza per ser de temps constant de còmput. Es a dir, per qualsevol imatge de la mateixa mida triga exactament ho mateix. Permet detectar canvis en la imatge que aproximadament ve a ser objectes. Finalment es trobar el centre de gravetat del núvol de punts després de passar-los per un threshold. Aquest centre de gravetat indica on esta la mitja dels objectes respecte la càmera, que permet orientar el robot. Per determinar si realment es un robot cal més criteri com per exemple trobar un mínim de punts.

3.4 Resultats

Als combats de sumo definitius es va veure com el robot detectava bé el contrincant, però a l'hora de moure tenia problemes. Moltes vegades no s'aguantava sol i anava per terra. Tampoc va tenir gaires bons resultats quan atacava. Les raons eren principalment dues, el grau de llibertat extra i moviments poc fluids.

El grau de llibertat que tant de joc donava a les proves de moure va resultar ser un fallo important a l'hora de fer sumo. La raó principal ve a ser el fet de que pujava el centre de gravetat. Al tenir un centre de gravetat més alt el robot resultava ser molt menys estable en mig del combat. Tampoc resultava molt més estable quan intentava moure defensivament.

El fet de no dedicar prou temps als moviments tampoc va ajudar. El robot amb el desgast de la competició va anar perdent facultats i als combats això es va veure clarament. Aquest és un problema inherent al model de fer servir els moviments programats en robobàsic sense realimentació.

Per solucionar els dos problemes, hagués estat interessant fer servir un robot sense el grau de llibertat extra per fer el combat. Idealment fent servir cinemàtica inversa que seria possible amb l'ús de altres projectes com el Project Hydrozoa [2]. La detecció i l'algoritme de sumo en sí van demostrar ser força fiables, però no van poder compensar els problemes mecànics.

4 Conclusions

El robot TWIKI va ser un dels millors a les proves de mobilitat. La segona posició darrera els vigents campions van ser va ser merescuda i fruit de molt esforç els dies anteriors. Si haguéssim tingut una comunicació més ràpida entre el gumstix i el microcontrollador, podríem haver-ho fet molt millor, però aquest projecte no estava terminat.

A la prova de sumo ens va sortir tot malament. Quedar penúltims va ser una sort, ja que el TWIKI semblava que anava a caure en qualsevol moment. El fet de que el 50% de la puntuació venia de la competició de sumo no ens va afavorir.

Globalment estem molt satisfets ja que el TWIKI fa ser fidel a la seva filosofia de intentar comportar-se com un humà. Els algoritmes van funcionar prou bé i el robot va respondre molt bé. S'ha pogut veure coses a millorar per intentar millorar la posició en els anys següents.

A Primer Apèndix: Usant l'entorn bitbake

A.1 Repositoris GIT

La filosofia de treball de l'equip del *Humanoid Lab* inclou usar els repositoris de software que posa a la nostra disposició. Un repositori no es res més que un directori, remot o local que gestiona els canvis que realitzem en el nostre programa.

A la wiki [3] podem trobar tota informació necessària per aprendre a usar els repositoris.

L'estructura actual dels repositoris es:

archive.git Legacy read-only files from the subversion

bioloid.git Bioloid Projects

bitbake-user-collection.git Bitbake recipes for Humanoid Lab

crobot.git crobot c++ repository

hydrozoa.git Reprograming micro-controller project

raul-pfc.git Raul Sanchidrian PFC

robots.git Demos and competitions final working code

simulator.git Crobot Simulator Repository

testing.git Testing repo for humanoid lab members

vision.git Vision Algorithms

Per treballar dins l'entorn bitbake usarem el bitbake-user-collection.git i el robots.git.

Per aprendre més coses dels repositoris GIT mireu: http://apollo.upc.es/humanoide/trac/wiki/git

A.2 Preparant l'entorn: Gumstix-oe enviroment

Primerament cal preparar l'entorn de cross-compilació. Tenim dos documents principalment de referencia:

- La wiki del projecte. [4]
- La web de documentació del gumstix. [1]

A.3 El nostre primer programa.

La primera tasca es batejar el nostre robot. El seu nom ens servirà d'identificador en diverses ocasions. La primera serà dins del repositori de robots.

A.3.1 Repositori robots.git

Per començar ens descarregarem el repositori robots.git des de http://apollo.upc.es/gitweb/.

Creem un nou directori amb el nom del nostre robot. En aquest punt podem crear un projecte des de zero, o bé copiar un d'existent i modificar-lo. La segona opció sempre serà més senzilla i ràpida. Queda a la vostra elecció.

Un cop creat el directori cal afegir-lo al repositori.

A.3.2 Repositori bitbake-user-collection.git

Si heu seguit la guia [4], procediu al seguent apartat. Si no l'heu seguit, aneu a [4] i feu tot el pas: **New packages to gumstix-oe**.

A.3.3 Afegint la nostra recepta.

En aquest punt, ja tenim el nostre repositori configurat, i el nostre entorn de treball bitbake també a punt.

Ara cal crear una nova recepta per indicar a bitbake com ha de compilar el codi del nou robot.

Aneu al vostre directori *user.collection/packages*. Creeu un nou directori anomenat *libcrobot-$NomDelRobot$*. Creeu dins un fitxer anomenat *libcrobot-$NomDelRobot$.bb*.

El seu contingut ha de ser similar a:

```
DESCRIPTION = "El meu algoritme de combat"
LICENSE = "GPL"

DEPENDS = "libcrobot"
## Descomenta la linea inferior si el teu codi usa la camara.
#DEPENDS = "opencv"

# Do not use the SRC_URI without tag or bitbake will crash
SRC_URI = "git://apollo.upc.es/git/robots.git;protocol=http;tag=master"

PACKAGES = "${PN}-dbg ${PN}"

FILES_${PN} =" \
        /usr/bin/*"

FILES_${PN}-dbg = "\
        /usr/bin/.debug/*"

PV = "git-${SRCDATE}"
S  = "${WORKDIR}/git/NomDelRobot"

# Home made hack to avoid stupid bitbake to search mirrors for the code
# It's necessary to pacth bitbake with this patch:
# https://lists.berlios.de/pipermail/bitbake-dev/2009-April/000430.html

SRC_TARBALL_STASH_${PN} = "none"

CFLAGS_append  = " -I${STAGING_INCDIR}"
LDFLAGS_append = " -L${STAGING_LIBDIR}"

do_compile() {
        cd ${S}
        oe_runmake
}
```

```
do_install() {
        oe_runmake DESTDIR="${D}" install
}
```

nota: aquest fitxer es similar al fitxer libcrobot-twiki.bb que ja es troba al repositori. Es recomana copiar i modificar el citat fitxer.

A.3.4 Cross-compila el teu primer programa.

Si has seguit bé tots els passos anteriors, pots compilar el teu programa fent:

```
$ bitbake libcrobot-NomDelRobot
```

Si tot ha anat bé, hauries de tenir un nou fitxer amb extensió *.ipk* a punt per ser enviat al Gumstix al directori *$GUMSTIX_OE/tmp/deploy/glibc/armv5te/* .

A.4 Dubtes

Si en qualsevol dels punts anteriors teniu qualsevol dubte, no dubteu en posar-vos en contacte amb algun dels membres administradors del projecte.

A.5 Petit Guió

Per als nous ubuntu's:

```
sudo rm /bin/sh
sudo ln -s /bin/bash /bin/sh

bitbake -c rebuild libcrobot-NomDelRobot

sudo rm /bin/sh
sudo ln -s /bin/dash /bin/sh

scp ${GUMSTIX_OE}/tmp/deploy/glibc/armv5te/\
libcrobot-NomDelRobot_git*.ipk root@192.168.1.X:

echo "done!"
```

References

[1] gumstix team. Setting up a build environment. http://www.gumstix.net/Setup-and-Programming/view/Getting-started/Setting-up-a-build-environment/111.html, 2010.

[2] J. Pegueroles, E. Simó. Hydrozoa project. http://apollo.upc.es/gitweb/?p=hydrozoa.git;a=summary, 2010.

[3] The Humanoid Lab Team. The Humanoid Lab Website. http://apollo.upc.es/humanoide/, 2010.

[4] The Humanoid Lab Team. The Humanoid Lab Website. http://apollo.upc.es/humanoide/trac/wiki/gumstixInstallingSoftware, 2010.

Acknowledgements

Gràcies a tot l'equip de *Humanoid Lab* [3] per fer possible la participació en el concurs.

IRI reports

This report is in the series of IRI technical reports.
All IRI technical reports are available for download at the IRI website
`http://www.iri.upc.edu`.