Resum

El treball que es presenta a continuació consisteix en el plantejament, disseny i construcció d'un robot paral·lel amb 6 graus de llibertat del tipus Gough-Stewart per controlar, mitjançant un control PID, la posició d'una bola situada a sobre de la seva plataforma mòbil. La detecció de la bola s'ha realitzat mitjançant un sistema de visió, amb una càmera en posició zenital enfocant la part superior de la plataforma mòbil del robot. S'ha procurat assegurar una resposta ràpida, amb la màxima precisió possible per part del robot, i fins i tot atractiva ja que el conjunt es vol fer servir per diferents jocs interactius.

El projecte involucra múltiples àrees de coneixement i precisa de l'ús d'una gran diversitat d'eines de programació, de càlcul i de construcció. El disseny del llaç de control, i fins i tot la pròpia estructura del projecte a nivell informàtic, s'ha realitzat amb Simulink® donada la seva capacitat per programar sistemes dinàmics directament a nivell de diagrames de blocs. En la programació de les rutines incloses en l'esquema de blocs s'hi troba codi programat en llenguatge Matlab® i, amb l'ajut de l'eina *Legacy code*, s'hi han inclòs blocs amb rutines que fan servir llibreries de C i C#. Aquestes llibreries contenen les eines de comunicació amb els motors a través d'un adaptador de port sèrie USB2Dynamixel. S'han emprat els motors de contínua Dynamixel^{TM} AX-12+ de tipus *servo*, que inclouen una memòria i un control intern programable. El procés de visió també fa servir eines de Matlab® i codi en aquest mateix llenguatge. Per una programació àgil de les transformacions en coordenades homogènies s'ha emprat les possibilitats de càlcul simbòlic de Matlab® i la llibreria "*Robotics Toolbox*". Els apartats corresponents al disseny i construcció del prototipus i del robot final, així com de l'estructura de suport de la càmera, han incorporat l'ús del programa de disseny assistit per ordinador SolidWorks® per així generar models que poguessin posteriorment ser impresos a les impressores 3D del laboratori.

Finalment, l'ajustament del controlador mitjançant el mètode de Ziegler-Nichols pel llaç tancat, amb un conjunt de refinaments, va fer possible la resposta desitjada. S'ha arribat així a un prototipus que compleix els requeriments que es varen establir inicialment i que, en permetre ajustar l'eix instantani de gir de la plataforma mòbil –tret que no era present inicialment– és capaç de fer traçar trajectòries arbitraries a la pilota. Aquests resultats aplanen el camí per fer servir aquest tipus de robots paral·lels en tot un seguit de aplicacions com ara la estabilització de sistemes subjectes a moviments arbitraris, jocs interactius, etc.

Aquest treball s'ha realitzat al laboratori de robots paral·lels de d'Institut de Robòtica i Informàtica Industrial a l'edifici de la Facultat de Matemàtiques i Estadística de la UPC.

Sumari

RESUM 1

SUMARI 3

1. OBJECTIUS DEL PROJECTE I ABAST 5

1.1. Definició del problema. Especificacions 5

1.2. Abast del projecte 6

2. ESTAT DE L'ART 8

2.1. Aportació 10

3. EINES 11

3.1. El software 11

3.1.1. Simulink® 11

3.1.2. Matlab® 12

3.2. El hardware 13

3.2.1. Unitat de procés central 13

3.2.2. Càmera 13

3.2.3. Actuadors 14

4. COMUNICACIÓ ORDINADOR-ACTUADORS 17

4.1. La connexió física 17

4.2. Els paquets de dades 18

4.2.1. Paquets d'instrucció 18

4.2.2. Paquets d'estat 20

4.3. La llibreria Dynamixel™ SDK 21

4.4. Funcions de comunicació 21

4.4.1. Funcions 22

4.4.2. Problemàtica sorgida 25

4.4.3. Legacy Code 26

5. VISIÓ 31

5.1. Imatges digitals 31

5.1.1. Espais de color 31

5.1.2. Matrius d'imatge 33

5.2. Muntatge 34

5.3. Processat i anàlisi de les imatges 36

5.3.1. Binarització 36
5.3.2. Esquema de visió de Simulink® 38
5.4. Tractament de les dades 42
5.4.1. Relacions entre els diferents sistemes de referència 43
5.4.2. Intersecció de la recta i el pla 45
5.4.3. Implementació en Simulink® 47

6. EL ROBOT 49
6.1. La plataforma de Gough-Stewart 49
6.1.1. Definicions 49
6.1.2. Avantatges 50
6.1.3. Arquitectura 51
6.2. Cinemàtica del robot 54
6.3. Càlcul de solucions 58
6.4. Programa de càlcul i simulador 65
6.4.1. Robotics toolbox 66
6.4.2. Pseudocodi 66
6.4.3. Simulador 69

7. EL CONTROL 70
7.1. Estratègia 70
7.1.1. Actuació en coordenades polars 70
7.1.2. Rotació entorn a dos eixos 72
7.2. Arquitectura del llaç 75
7.2.1. Controladors PID 75
7.2.2. Sintonía dels PIDs 77
7.2.3. Sintonía i ajustament 79
7.3. El llaç en Simulink® 81

8. RESULTATS 87

CONCLUSIONS I TREBALL FUTUR 91

AGRAÏMENTS 93

BIBLIOGRAFIA 94
Referències bibliogràfiques 94

1. Objectius del projecte i abast

L'objectiu d'aquest projecte consisteix en crear un robot paral·lel que, actuant sobre una plataforma i amb l'ajut d'un sistema de visió, controli la posició d'una bola que rodoli, o que fins i tot boti, per la seva superfície. L'ús d'un robot paral·lel de 6 graus de llibertat per aquesta tasca permet augmentar la capacitat de control sobre la pilota, atorgant-li una major versatilitat. D'aquesta manera tenim control sobre la localització de l'eix instantani de rotació de la plataforma mòbil, la qual cosa és impossible amb un robot de, per exemple, dos graus de llibertat.

Degut a la gran component d'integració de múltiples àrees en aquest projecte, a continuació s'exposa un desglossament dels elements fonamentals per assolir aquest objectiu.

1.1. Definició del problema. Especificacions

En les diferents seccions clarament diferenciades entre sí d'aquest projecte, es pot definir la problemàtica enfrontada. En cada secció es descriuen els punts clau de l'apartat en qüestió.

Actuadors

- Els actuadors triats han d'incorporar un sistema per poder-ne controlar l'angle rotat, parell i velocitat, i que la comunicació amb ells no suposin un coll d'ampolla per l'objectiu global.

- La comunicació amb el controlador de cada actuador ha de ser no només ràpida sinó també fàcil de programar des del software de control triat.

Sistema de visió

- La càmera triada ha de donar diferents solucions pel que fa a resolució de la imatge i a fotogrames per segon de vídeo.

- El processat d'imatges és altament costós en recursos de la unitat de càlcul informàtic. Cal agilitzar en la mesura del possible aquesta etapa.

- La situació de la càmera ha de poder donar informació al robot en tot l'espai de treball d'aquest.

- Cal identificar la pilota a la imatge.

- És necessari desenvolupar l'algorisme per extreure la informació rellevant de la imatge processada.

El robot

- L'arquitectura del robot ha de permetre una velocitat de canvi d'orientació i posició suficient per ajustar-se a la resposta desitjada de la pilota. L'espai de treball i les configuracions accessibles han de ser suficients per l'objectiu general del projecte.

- És necessari triar la manera en què es vol que el robot es mogui per tal de controlar la localització de la pilota.

- La programació d'una rutina de càlcul del moviment del robot (en altres paraules, de càlcul de la seva cinemàtica inversa) és necessària ja que el control haurà d'actuar contínuament sobre la pilota.

El tipus de control

- El control ha de permetre mantenir la pilota en la posició desitjada minimitzant-ne l'error.

- La pilota ha de poder seguir trajectòries programades de manera controlada.

- Cal idear una estratègia de control per enviar instruccions al robot en funció de la informació que rep del sistema de visió. Per tant, triar quines són les variables de control.

1.2. Abast del projecte

En aquest projecte s'exposen les diverses etapes fins l'assoliment de l'objectiu, amb una definició dels conceptes amb els quals cal un coneixement o familiarització per enfrontar-s'hi. També s'hi inclou una descripció del software i el hardware necessaris per dur a terme la tasca plantejada amb la justificació corresponent o notes sobre alternatives estudiades.

És molt important destacar que el treball ha mantingut una visió pragmàtica. Això ha comportat prioritzar l'assoliment de l'objectiu final amb el material i el temps a disposició. D'aquesta manera, s'ha restringit el nivell d'exhaustivitat de cada apartat per poder avançar durant la realització del projecte sense renunciar als objectius principals. Per tots aquests motius, no es varen establir especificacions per a la resposta del la pilota en la plataforma pel que fa a precisió, temps d'establiment, velocitat màxima controlable per la plataforma, entre d'altres.

A continuació es defineix l'abast del projecte per apartats:

La comunicació amb actuadors

Aquesta fase ha de treballar sobre el software proporcionat pel fabricant per cobrir les necessitats de la comunicació. També ha d'incloure la integració amb el software de control.

La visió

Ha de ser capaç de proporcionar al control informació sobre la posició de la pilota en tot l'espai de treball del robot mitjançant la implementació d'algorismes de processat i anàlisi d'imatges. Dat el fet que el processat i anàlisi d'imatges genera un alt consum de recursos informàtics en comparació amb altres sistemes de sensat (per exemple els de contacte), l'algorisme haurà de tendir a la simplicitat. De la mateixa manera, el tipus de càmera, la resolució de la imatge i la velocitat de presa d'aquestes s'han d'ajustar per garantir una velocitat d'actuació adequada pel control.

El robot

En aquest apartat cal realitzar l'elaboració d'un disseny propi i la construcció del robot paral·lel amb la velocitat i mida adequades per la tasca. També cal realitzar el disseny i muntatge de la plataforma mòbil.

Cal tractar també l'estudi de la cinemàtica inversa del robot per poder traduir els canvis de posició i orientació de la plataforma mòbil en moviments dels actuadors.

El control

El llaç de control es programara en Simulink®, degut a la seva interfície entenedora basada en diagrames de blocs i la seva integració total amb el programari de Matlab.

S'inclou en aquest treball el disseny de l'estratègia de control (nombre de controladors, variables de control, etc) que obtingui els resultats més satisfactoris.

El control es vol realitzar amb controladors del tipus PID per la seva simplicitat, ajustant-ne els paràmetres amb la metodologia més adient al problema que ens ocupa. Es vol el control que presenti la resposta més atractiva possible pel que fa a velocitat de resposta, temps d'establiment i precisió. També es vol minimitzar el sobrepuig en la resposta.

2. Estat de l'art

En la literatura disponible sobre el control de la posició d'una bola sobre un pla, el més comú és trobar articles relacionats amb el disseny del controlador per plataformes amb només 2 graus de llibertat. Les plataformes emprades sovint són com les de la figures 2.1 i 2.2. Aquests sistemes són els dominants degut a la ja complicada tasca del control de la pilota i l'estimació de la seva dinàmica, que fa que es recorri al prototipus de plataforma amb model dinàmic conegut.

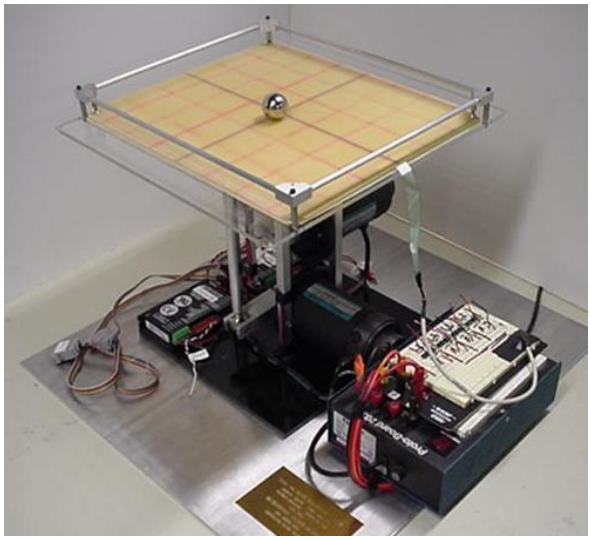


Fig. 2.1. *Plataforma de 2 graus de llibertat. Adaptat de [1]*

Els sistemes de balanceig són uns dels problemes més desafiants en la teoria de control. Existeixen moltes plataformes i controls proposats per aquests problemes de control en 2D: pèndol invertit, bola sobre carril i fins i tot dobles i triples pèndols invertits. L'equilibratge d'una pilota sobre el pla es una versió estesa del problema tradicional d'una bola rodolant lliurement sense lliscar per un carril. El sistema a controlar consisteix en una plataforma que pot ser manipulada en dues direccions perpendiculars. L'objectiu general d'aquests treballs i articles ha consistit en moure la pilota d'una posició a una altra de la plataforma, mantenir-la quieta i estable en la posició o bé recórrer trajectòries programades. En aquests problemes de control d'estabilització es procura minimitzar l'error i el temps d'establiment de la resposta. Aquest tipus de plataformes solen estar presents en laboratoris de control i sovint és fan servir per verificar la *performance* i la efectivitat de nous algorismes de control. Els controladors trobats en la literatura, tal com veurem a continuació, poden estar tan basats en un model matemàtic simplificat de la dinàmica del sistema com no. En aquest darrer cas, el controlador de vegades es diu que és *lliure*.

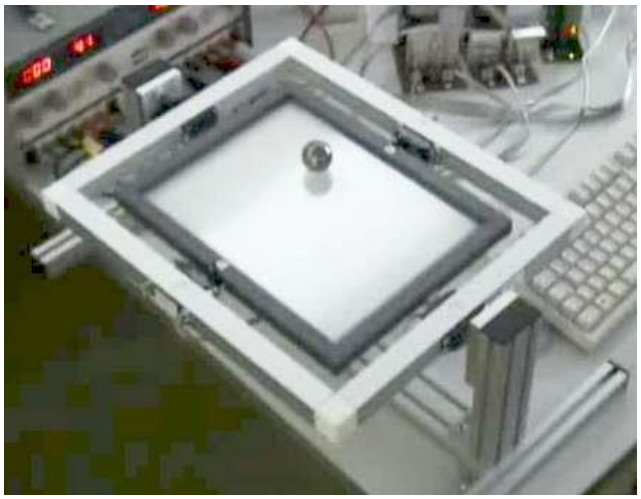


Fig. 2.2. *Un altre model de plataforma de 2 graus de llibertat. [2]*

Múltiples aspectes del disseny mecànic i electrònic, la implementació i el control del sistema s'han investigat en la literatura existent [1, 3-11]. En aquests estudis, el disseny del control s'ha fet a partir del model simplificat de les equacions de la dinàmica del sistema de la bola rodant sobre el pla, i el desenvolupament d'un model dinàmic detallat ha quedat en segon pla.

En l'article [1], basat en el model linealitzat respecte el punt d'equilibri, es dissenya un controlador PID per estabilitzar el sistema. En els articles [3] i [8] es considera un model no lineal amb un control de canvi d'estat que es fa servir per poder aconseguir que la bola segui les trajectòries desitjades. Pel que fa a [11], es dissenya un controlador PID amb un observador.

Pel sistema bola i plataforma, els sensors per detectar la posició de la pilota es poden dividir en dos grans tipus: els basats en una plataforma tàctil [1,11] i els basats en una càmera amb una visió zenital [3-7,9,10]. Ambdós casos presenten certes mancances. La pilota fàcilment pot perdre contacte amb la superfície durant el moviment, fet que causa discontinuïtats en la mesura de la posició si es fa servir un sensor de tacte. Per altra banda, si es fa servir una càmera, degut al baix període de mostreig (sol estar entorn als 30Hz), poden aparèixer limitacions de la resposta en temps-real del sistema.

El *visual-servoing* [12,13,14] fa referència a una línia de recerca i desenvolupament tecnològic a on s'incorpora informació visual als llaços de control. L'adquisició d'imatges en temps real i el processat d'aquestes és un punt crític en aquest tipus d'aplicacions. Un número alt de fotogrames per segon i un temps de processat baix són crucials degut al curt temps que té el sistema per prendre decisions, en general, sobre l'escena observada en cada període de mostratge. El processat d'imatges en temps real requereix un ritme de processat de píxels elevat, computació paral·lela i un ús eficient del hardware. Avui en dia, degut a l'arquitectura seqüencial dels processadors moderns, es tendeix a l'ús d'FPGAs que presenten una flexibilitat i potència de càlcul en paral·lel molt superior [15,16].

Finalment, pel que fa al camp del control de bola sobre el pla amb plataformes paral·lels del tipus Gough-Stewart, només s'ha trobat la feina feta per un grup que treballa amb aquests robots per a simuladors de vol i dissenyà un control PID [17] en una línia semblant a la aquí tractada. Tot i que no se'n donen gaire detalls, en aquest cas es fa servir també un panell tàctil per detectar la posició de la pilota.

2.1. Aportació

En aquest projecte, doncs, es tria per dotar de major complexitat al moviment de la plataforma i donar més versatilitat al seu control. Gràcies als 6 graus de llibertat del robot, es pot situar l'eix de rotació instantani de la plataforma en 3 dimensions dins de l'espai de treball i augmentar el nombre d'estratègies de control empleades. Per poder abastir tots els objectius plantejats, es realitza un control desvinculat del model matemàtic i un *visual-servoing* per augmentar, per una banda, l'aspecte didàctic del projecte i, per una altra, per una raó purament econòmica. Es descriu el camí complet en la realització del sistema com a una tasca d'integració multidisciplinària incloent: processat d'imatges digitals, cinemàtica de robots paral·lels, disseny de controladors PID, i comunicacions digitals.

3. Eines

Una vegada definits els objectius i havent revisat la literatura existent, va caldre definir les eines de treball, comunes durant tot el desenvolupament del projecte. Essencialment es dividiren en dues categories essencials: el software i el hardware. Els productes finalment presents al projecte no només tenen una estreta relació amb el laboratori de cinemàtica de l'IRI sinó que, a més a més, presentaven una sèrie d'avantatges que seran descrits a continuació. L'àmplia disponibilitat del material, les llicències, el contacte amb els proveïdors i l'experiència del grup del laboratori van ser decisives en la tria. Es va assegurar la capacitat d'aquestes eines per assolir els objectius proposats.

3.1. El software

Per la natura del projecte, es va decidir utilitzar Matlab® i Simulink® per tot el referent al processat de la informació provinent de la visió i la comunicació amb els actuadors. També va caldre SolidWorks® com a programa de CAD per poder dissenyar les peces necessàries per construir el robot.

Cal esmentar que el projecte treballa a baix nivell amb el *Software Development Kit* de Dynamixel™ en llenguatge C i C# per la comunicació amb els actuadors, al qual s'hi van afegir modificacions.

3.1.1. Simulink®

És el software central entorn el qual gira el projecte. Simulink® és un entorn gràfic interactiu per al modelat, anàlisi i simulació de gran varietat de sistemes dinàmics (discrets, analògics i híbrids) mitjançant la utilització de diagrames de blocs. Permet la incorporació d'algorismes i controls que s'hagin desenvolupat en C prèviament. Treballa totalment integrat amb Matlab®. Aquest software va proporcionar uns trets diferencials:

- L'àmplia presència que té en el món de la docència i de la investigació tecnològica actual i la robòtica.
- La seva versatilitat, compatible amb components i dispositius externs.

- La gran varietat de llibreries que li atorguen potencia i alt nivell.

- La seva interfície intuïtiva que permet fer modificacions per blocs funcionals, planificació d'estructures i compartició d'esquemes altament intel·ligibles. Aquest fet en permet l'ús demostratiu i la compartició amb gent aliena als models per tota mena d'ampliacions i de retocs.

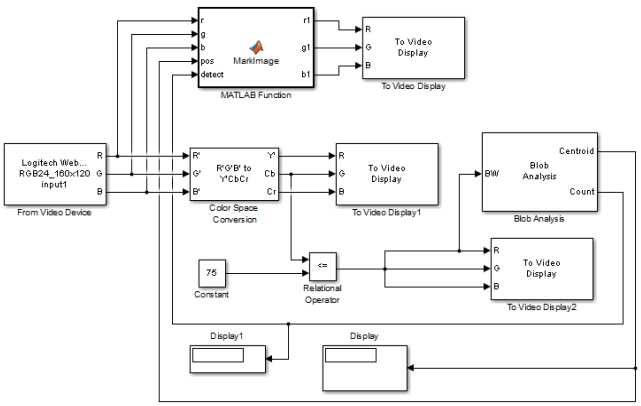


Fig. 3.1. *Exemple d'un esquema Simulink® de pel processat i anàlisi de les imatges procedents d'una càmera*

3.1.2. Matlab®

Matlab® és un entorn de computació numèrica i un llenguatge de programació. Creat per la companyia MathWorks®, Matlab® permet manipular fàcilment matrius, dibuixar funcions i dades, implementar algorismes, crear interfícies d'usuari, i comunicar-se amb altres programes en altres llenguatges. Tot i que s'especialitza en computació numèrica, una caixa d'eines opcional (*toolbox*) permet usar el motor de càlcul simbòlic de Maple™.

El programa final, tot i ser estat executat i implementat dins d'un entorn de Simulink®, inclou diversos blocs programats en llenguatge Matlab®. Ambdós entorns pertanyen al mateix desenvolupador i es complementen l'un a l'altre atorgant una potència superior que per separat. L'ús de l'entorn de Matlab® va permetre:

- La familiarització amb la llibreria de baix nivell dels motors.

- La programació i realització de proves de manera àgil sobre una entrada de línia de comandes.

- L'organització ràpida i còmoda de scripts de prova.
- L'ús com a eina de càlcul simbòlic i numèric molt potent.
- La utilització d'aplicacions molt útils ja definides com ara l'anàlisi d'imatges binaritzades.

3.2. El hardware

El hardware emprat en aquest projecte es troba dividit en els diferents elements que participen en el llaç de control. Aquests elements corresponen a una unitat de procés central (CPU), un sensor de presa d'imatges i, finalment, els actuadors.

3.2.1. Unitat de procés central

La unitat de procés central és l'element encarregat de realitzar tots els càlculs i la comunicació entre els altres elements del llaç de control. Podria ser qualsevol dispositiu amb una CPU amb prou capacitat per gestionar la informació a la velocitat necessària per garantir el control. Degut al fet que tot el processat d'imatge requereix de potència, una placa del tipus Arduino® o Rapsberry Pi® podien mancar aquesta potència i la complexitat de la implementació augmentava. Finalment es va decidir implementar tot el codi i els programes en un ordinador personal convencional amb les especificacions tècniques següents:

- Sistema operatiu: Windows 7 professional 64bits
- Processador: Intel® Core(TM) 2 Quad CPU Q9650 @ 3.00GHz (cada nucli)
- Memòria RAM: 8.00GB

Aquestes especificacions han provat ser suficients per l'aplicació desitjada.

3.2.2. Càmera

És clar que pel control de qualsevol sistema de llaç tancat que tingui per sortida una actuació sobre el món físic i que es vulgui comparar amb una magnitud desitjada no es podrà prescindir d'un sensat d'aquesta resposta física. Dat que el sistema d'estudi havia de ser capaç de situar la pilota en diferents punts de la plataforma i de partir de configuracions inicials diferents, era necessari triar un mètode que permetés al control conèixer el comportament de la pilota. D'aquesta manera, avaluant els sistemes possibles per localitzar la pilota van aparèixer dues possibilitats: fer sevir un panell tàctil o un sistema de visió. Ambdós solucions són molt comunes al món de la robòtica però, donada la versatilitat de la

visió i el seu baix cost, es va acabar triant aquesta darrera opció.

La càmera triada va ser de la casa Point Grey, de la línia de Grasshopper i, concretament, el model Grass-03K2C-C (veure figura 3.2). Se'n poden trobar les especificacions completes a l'annex. Aquestes càmeres amb aplicacions industrials presenten una sèrie d'avantatges davant d'una web cam convencional, que es presentava com una altra alternativa viable:

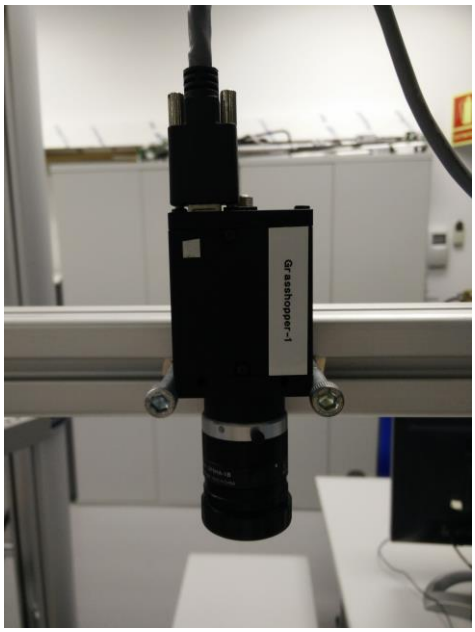


Fig. 3.2. *Fotografia de la càmera Grasshopper emprada en el projecte.*

- El tret diferencial més important és la velocitat de comunicació entre el dispositiu perifèric i l'ordinador. Aquestes càmeres usen el que es coneix com a *Firewire*. El *Firewire* és un tipus de comunicació via cable entre el perifèric i CPU que exigeix d'una placa de comunicacions específica connectada a la placa base de l'ordinador. Permet un ritme de comunicació superior al el de l'USB convencional (fins a 800 Mbits/s) i, a més a més, no consumeix recursos del processador en enviar la informació, ja que disposa de la seva pròpia placa. En un projecte on es tracte d'assolir el cicle de mostreig més alt possible, aquesta fou la característica principal per triar aquest hardware.

- Resolucions superiors a la d'una web cam sense deformació de la imatge, ja que la lent de les primeres deforma cap als marges.

- Alta velocitat (fins a 200fps).

- Paràmetres editables, la qual cosa permet múltiples configuracions diferents.

3.2.3. Actuadors

Els actuadors de Dynamixel^{TM} de la sèrie AX i tipus AX-12+ foren els utilitzats, presents en molts robots hominoides de baix cost i amb una alta versatilitat. Aquests actuadors són de fet servomotors, els quals es distingeixen dels motors de contínua comuns per la capacitat de situar el valor de l'angle relatiu entre rotor i estator mitjançant un codificador rotacional i un controlador integrat.

El codificador rotacional consisteix en un dispositiu electromecànic emprat per convertir la posició angular de l'eix en codi digital. El controlador, per altra banda, consisteix en una memòria estàtica del tipus EEPROM i una memòria dinàmica RAM les quals s'encarreguen d'indexar i emmagatzemar les dades de l'estat actual del servomotor. Cada adreça d'aquesta memòria és un byte i molts dels paràmetres consisteixen en paraules de dos bytes. En aquest cas la paraula es troba en dues adreces consecutives.

Fig. 3.3 *Servomotors AX-12+ vista frontal i vista del darrere*

Mitjançant la modificació de paràmetres d'aquesta taula es pot modificar el comportament del motor. Aquesta modificació es realitza via un protocol de comunicació establert pel fabricant, amb un adaptador de port sèrie. La comunicació es realitza amb l'enviament de paquets de dades a un ritme fixable però acotat de bits per segon.

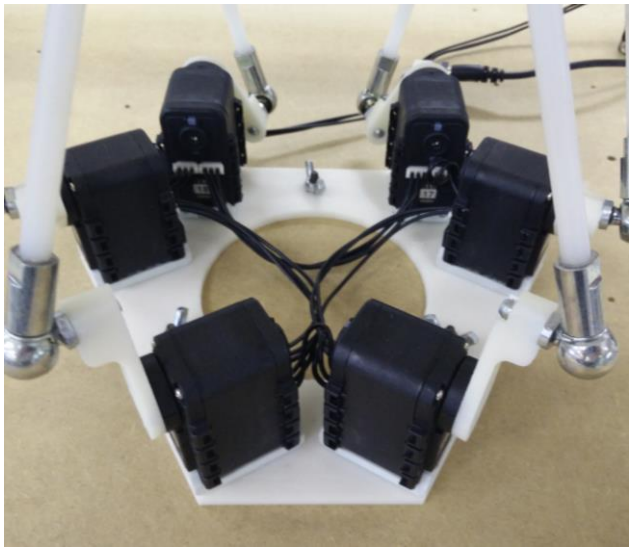


Fig. 3.4 *Connexió en* daisy-chain *dels motors al prototipus final*

Un avantatge d'aquest protocol de comunicació dissenyat per Dynamixel™, essencial a l'hora d'analitzar-ne la funcionalitat, és la característica del que s'anomena comunicació en *Daisy-Chain*, consistent en ser capaç d'escriure alhora a les memòries d'una cadena de motors connectats en sèrie (veure figura 3.4) .Aquest fet és favorable sempre que calgui executar composicions de moviments en un mateix interval de temps, o iniciar-los en un mateix instant. D'altra banda, caldria escriure als actuadors un per un amb el retard temporal que això comportaria, ja que fins que no s'envia el paquet i es retorna l'estat de la comunicació, no queda el canal lliure altre cop. Més endavant en aquesta memòria es tracta la comunicació computador-actuadors amb detall. Per l'ús que es fa d'aquests motors en aquest projecte, on es controlen múltiples actuadors alhora, aquest tret es fa imprescindible.

Les especificacions dels motors, així com l'estructura de les taules de control i les adreces dels paràmetres i les instruccions es troben a l'annex.

4. Comunicació ordinador-actuadors

En aquest apartat s'analitzen en detall tots els elements de la comunicació entre els servomotors i la central de procés, en aquest cas un PC. Es presenten els elements en ordre de més baix a més alt nivell, en ordre lògic de familiarització i programació de la comunicació.

4.1. La connexió física

L'adaptador USB2Dynamixel (figura 4.1.) és l'utilitzat en totes les proves realitzades i el model final. Aquest adaptador de port sèrie tipus USB a TTL (3 pins) / RS485 (4 pins) / RS232 (port sèrie) funciona amb el protocol de comunicació 1.0 de Dynamixel™.

Fig. 4.1 *Adaptador USB2Dynamixel utilitzat*

Un port sèrie consisteix en un sistema de comunicació entre màquina i perifèrics per un canal de transmissió on les dades s'envien bit a bit, a diferència d'un port paral·lel on hi ha diversos canals de comunicació. Un exemple d'aquest tipus de port és el port USB, i aquest adaptador converteix les dades que arriben pel cable USB estàndard de l'ordinador en dades que surten per un port TTL (*Transistor-Transistor Logic*), que és el tipus de connexió que exigeixen els motors. La sortida TTL consisteix en tres pins: alimentació, terra i dades (figura 4.2). Aquests pins es troben connectats amb el seus anàlegs, de manera que el control es pot realitzar amb un únic cable d'entrada connectat, o bé construir una cadena en sèrie de motors on tots reben el mateix paquet de dades (la connexió en *Daisy Chain* esmentada anteriorment). Aquests paquets, contenint les dades adients, son llegits només pels motors d'interès.

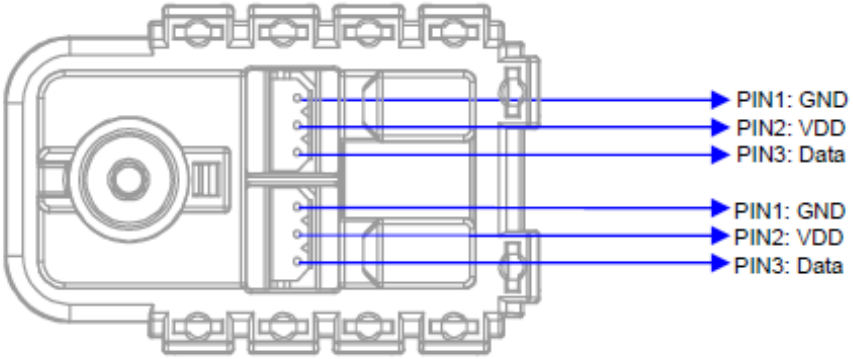


Fig. 4.2. *Esquema de distribució de pins en un motor AX-12+ [18]*

4.2. Els paquets de dades

Els motors i el computador es comuniquen mitjançant un sistema de paquets de dades enviats de forma bidireccional. Aquests paquets de dades consisteixen una seqüència de bits amb una estructura determinada que és reconeguda pel maquinari tant dels motors com de l'ordinador.

En el cas que ens ocupa, el controlador principal (ordinador) envia paquets que, en arribar al xip de l'actuador destí, aquest en rebota un paquet d'estat de la connexió. Els paquets que envia l'ordinador es denominem *Instruction Packets* (paquets d'instrucció) i els paquets que retornen, si calen, s'anomenen *Status Packets* (paquets d'estat).

Cal esmentar que els paquets no deixen de ser seqüències de números en el temps expressats en la base que es desitgi per tal de fer-los intel·ligibles i, tot i que en última instància el llenguatge màquina és binari, normalment s'expressen en base hexadecimal.

4.2.1. Paquets d'instrucció

L'estructura general dels paquets d'instrucció és la següent:

0xFF 0xFF ID LONGITUD INSTRUCCIÓ PARÀMETRE 1 ... PARÀMETRE N *CHECKSUM*

Recordem que els números en hexadecimal van precedits d'una capçalera que ens indica la base "0x". Cada instrucció de dos números hexadecimals correspon a un byte d'informació, és a dir 8 bits o bé un valor decimal comprès entre 0 i 255.

El significat de cada byte que conforma el paquet és el següent:

0xFF 0xFF → 2 bytes que indiquen el principi del paquet, en són la capçalera.

ID → Es el número identificador del motor al que va dirigit el paquet. És un número comprès entre 0 i 253 (0x00 i 0xFD). Es reserva el número 254 per la instrucció enviada a tots els motors d'una cadena. Quan això passa, tots reben la instrucció i no es retorna un *Status Packet*.

LONGITUD → És la longitud del paquet en bytes. Aquesta longitud es calcula com "el nombre de paràmetres que conté + 2". El 2 correspon a la ID i la LONGITUD.

INSTRUCCIÓ → Aquesta comanda dóna una instrucció a Dynamixel™ i té els tipus següents definits pel fabricant, el que vol dir que són fixes. (Vegeu taula 4.1)

Valor	Nom	Funció	Nº de paràmetres
0x01	PING	Cap execució. Comprova la connexió.	0
0x02	READ_DATA	Llegeix dades de la memòria del motor.	2
0x03	WRITE_DATA	Escriu dades a la memòria del motor.	2 o més
0x04	REG WRITE	Com WRITE_DATA però espera una acció ACTION per executar-se.	2 o més
0x05	ACTION	Activa l'escriptura de dades del REG WRITE	0
0x06	RESET	Torna el xip integrat a paràmetres de fàbrica.	0
0x83	SYNC WRITE	Es fa servir per controlar diversos motors Dynamixel™ alhora.	4 o més

Taula 4.1. Conjunt d'instruccions dels paquets de comunicació.

PARÀMETRES 1...N → Aquests bits s'hi inclouen quan la instrucció del paquet requereix informació addicional.

CHECK SUM → Es fa servir per comprovar si un paquet ha sofert pèrdues d'informació

durant la comunicació. És algorisme integrat en les llibreries emprades.

CHECK SUM = ~(id + longitud + instrucció + paràmetre 1 + ... + paràmetre n)

En aquest cas el símbol "~" indica l'operador lògic de negació. Tot això resulta en un byte que es situa al final del paquet.

4.2.2. Paquets d'estat

Els software dels motors executa les instruccions rebudes en el paquet d'instrucció i en retorna el resultat al controlador principal (computador en aquest cas). L'estructura que presenta és la següent:

0xFF 0xFF ID LONGITUD ERROR PARÀMETRE 1 ... PARÀMETRE N *CHECKSUM*

La capçalera, el número identificador dels servomotors i la longitud és exactament igual. De la mateixa manera els paràmetres i el *checksum* mantenen les mateixes utilitats. Cal esmentar que els paràmetres en aquest cas aporten informació addicional en la comunicació.

ERROR → Retorna el tipus d'error durant la operació executada en un byte. El bit que correspon a cada tipus d'error es troba estipulat tal com indica la taula 4.2.

Bit	Nom de l'error	Contingut
7	0	-
6	Instrucció	En cas d'haver rebut una instrucció desconeguda val 1.
5	Sobrecàrrega	En cas d'excedir el parell màxim val 1.
4	*Checksum*	En cas pèrdua dades durant la comunicació val 1.
3	Rang	En cas d'intent d'accedir o escriure a un valor fora de rang val 1.
2	Sobreescalfament	En cas d'excedir el rang de temperatures d'operació val 1.
1	Límit d'angle	En cas d'escriure una posició no accessible val 1.
0	Voltatge d'entrada	En cas de detectar un voltatge massa alt/baix per operar val 1.

Taula 4.2. Taula de la correspondència de cada bit d'error del byte ERROR.

Exemples dels diferents paquets per cada instrucció i pels paquets d'estats poden ser trobats a l'annex

4.3. La llibreria Dynamixel™ SDK

El Dynamixel™ SDK o *Software Development Kit* és una llibreria de baix nivell en llenguatge C que Dynamixel™ posa a l'abast dels usuaris per donar un marc de treball menys rudimentari que el llenguatge màquina. Aquesta llibreria permet construir els paquets de comunicació hexadecimals encarregats de parlar amb un adaptador de port sèrie distribuït per l'empresa i també inclou alguns mètodes bàsics ja programats en un nivell d'abstracció superior. Sigui quin sigui l'entorn de programació acaba acudint a questa llibreria en C#.

Permet realitzar algunes de les operacions bàsiques com ara:

- Iniciar i finalitzar la comunicació.
- Construir els paquets de dades. Rebre, establir i enviar.
- Construir de forma directa els paquets per a operacions simples com ara una lectura d'un byte d'una memòria, escriure-hi, fer un *"ping",* etc.
- Actuar a nivell de sistema (obrir ports, interactuar amb els busos, passar a hexadecimal, ...)

La descripció de les funcions d'aquesta llibreria es pot trobar a l'annex.

4.4. Funcions de comunicació

Tota la familiarització amb la llibreria oferta pel fabricant així com amb l'estructura de paquets es va realitzar des de Matlab®. Aquest procés fou necessari degut a que treballar amb el robot paral·lel exigia funcions de més alt nivell que les predefinides.

En primera instància, calia una comunicació molt més ràpida i aquest fet implica una lectura i escriptura als motors realitzada alhora. Per tant, la creació d'una llibreria pròpia en llenguatge Matlab® podia agilitzar notablement el procés de treball amb els motors. Cal tenir present que aquesta només havia de ser l'eina còmoda a través de la qual era possible la comunicació amb els motors.

Es va trobar que la instrucció de lectura de tots els actuadors alhora no és present en aquest model concret d'actuador i finalment no es pogué implementar. Les funcions, doncs, que sí es van implementar en previsió de les necessitats de comunicació amb el robot s'especifiquen a continuació en pseudocodi així com la descripció dels mètodes ja existents que també participen en la comunicació d'aquest projecte.

La documentació referent a les rutines programades en codi original també s'inclou a l'annex.

4.4.1. Funcions

Inicialització

Aquesta funció és l'encarregada d'iniciar la comunicació entre computador i actuadors. És una funció existent en la llibreria Dynamixel™ SDK i precisa de:

- La definició d'una llista d'IDs dels motors connectats.
- El número d'un port de comunicació sèrie on s'hi ha connectat l'adaptador. Windows els reconeix com a ports COM.
- La velocitat de comunicació en baudis. Aquest tret ve donat pel tipus de motor que s'empri i simplement es fa servir per indicar a l'adaptador el ritme en què ha d'enviar les dades per tal que siguin rebudes correctament. Amb els motors AX-12+ aquest valor ve fixat a 34 baudis, el que correspon a 57142 bps.

Si la connexió s'ha iniciat amb èxit retorna un valor 1, i 0 en cas de fallada.

Terminació

Aquesta funció és l'encarregada de tancar la comunicació. No precisa de cap paràmetre ni retorna cap valor.

Escriptura sincronitzada

Aquesta funció va ser programada per tal d'escriure a tots els motors enviant un sol paquet d'instrucció. Com s'ha esmentat anteriorment, existeix la instrucció de paquet SYNC_WRITE i la funció desenvolupada construeix els paquets seguint l'estructura ja especificada de forma genèrica i a l'annex concretament. Precisa de:

- El nombre d'actuadors connectats en *daisy chain*.
- Un nombre enter en base decimal igual a l'adreça de la memòria, igual per tots els motors, on es comença a escriure.

- Una llista amb els identificadors (IDs) de tots els actuadors connectats entre sí.

- Una llista amb els valors en base decimal a escriure a cada actuador ordenats amb concordança amb l'anteriorment especificat. Un byte sovint no és suficient per representar tot el rang de valors que pot prendre un paràmetre i, en aquest cas, es calcula el valor hexadecimal d'un byte "baix" i un "alt" per cada valor decimal i es divideix en dos paràmetres que s'escriuen en cel·les de memòria consecutives. Aquest procés és automàtic en la rutina programada.

Cal afegir que durant la rutina s'executen funcions de la llibreria Dynamixel™ SDK per muntar paquets de dades. Aquestes funcions predefinides s'indiquen de color vermell traduïdes per fer-les més entenedores. El nom real s'afegeix al costat per qualsevol consulta que es vulgui fer sobre el seu funcionament. El pseudocodi corresponent a les accions que realitza és el següent:

```
Funció escriptura_sincronitzada (nombre_actuadors, adreça, llista_IDs,
llista_valors)

%Comprovem que calgui muntar el paquet
Si nombre_actuadors = 0 llavors
      retorna 0
Sinó si nombre_actuadors = 1 llavors
      Escriu_paraula( primera ID, adreça, primer valor) %dxl_write_word
      retorna 0
Fi Si

%Múltiples valors, creem el paquet amb les funcions predefinides
%ID per a enviar a tots els motors és 254 o BROADCAST_ID (0xFE)
Estableix_ID_de_paquet(254) %dxl_set_txpacket_id
%Instrucció de SYNC_WRITE és 131 (0x83)
Estableix_instrucció_de_paquet(131) %dxl_set_txpacket_instruction
%Adreça on comença a escriure dels motors
Estableix_paràmetre(0, adreça) %dxl_set_txpacket_parameter
%Longitud de cada valor a escriure (cada paraula = 2 bytes per tot el que
cal manipular)
Estableix_paràmetre(1, 2)

%Bucle per cada ID donada
Per i des de 0 fins a (nombre_actuadors -1) fes
      %Prendre la ID corresponent i el valor que li correspon
      Estableix_paràmetre(2+3*i, llista_IDs[i])
      %Muntem la seqüència de bytes, cada paràmetre és un byte i la
      paraula es de 2
      Estableix_paràmetre(2+3*i+1, calcula_byte_baix(llista_valors[i]))
      Estableix_paràmetre(2+3*i+2, calcula_byte_alt(llista_valors[i]))
Fi Per
```

```
%La longitud total
Estableix_longitud_paquet((2+1)*Nombre_actuadors+4)
%dxl_set_txpacket_length
%Enviem el paquet
Envia_paquet %dxl_txrx_packet

retorna Estat_comunicació %No hi ha paquet de retorn

Fi Funció
```

Escriptura sincronitzada de posició desitjada i velocitat de moviment

En previsió de la possible necessitat d'imposar velocitats de moviment a cada actuador dins del rang d'operació permès, es va dissenyar una altra rutina que ho permetés semblant a l'anterior. Fa servir la mateixa instrucció d'escriptura sincronitzada i, per tant, la mateixa estructura de construcció de paquet.

Aquesta funció varia de l'anterior en el següent:

- Precisa de dues llistes de valors ordenades segons les IDs on arriben, una per les posicions desitjades i l'altra per les velocitats.
- No precisa d'adreça d'escriptura. Aprofitant el fet que sempre s'escriurà en cel·les de posició i que consecutivament a aquestes s'hi troben les de velocitat de moviment (ja dissenyat expressament així pel desenvolupador) la rutina ja imposa l'adreça i la longitud de les dades a escriure (adreça de posició i longitud 4, 2 bytes de posició i 2 bytes de velocitat).

El pseudocodi corresponent de les accions que realitza és el següent:

```
Funció establir_posicions_velocitats(nombre_actuadors, llista_IDs,
llista_posicions, llista_velocitats)

%Comprovem que calgui muntar el paquet
Si nombre_actuadors = 0 llavors
      retorna 0
Fi Si

%Múltiples valors, creem el paquet amb les funcions predefinides
%ID per a enviar a tots els motors és 254 o BROADCAST_ID (0xFE)
Estableix_ID_de_paquet(254)
%Instrucció de SYNC_WRITE és 131 (0x83)
Estableix_instrucció_de_paquet(131)
```

```
%Adreça on comença a escriure dels motors
Estableix_paràmetre(0, 30) % L'adreça a la memòria sempre serà 30
%Longitud de cada valor a escriure (cada paraula = 2 bytes per tot el que
cal manipular)
%com que tenim 2 paraules (posició i velocitat) de 2 bytes cadascuna la
longitud és 4
Estableix_paràmetre(1, 4)

%Bucle per cada ID donada
Per i des de 0 fins a (nombre_actuadors -1) fes
      %Prendre la ID corresponent i el valor que li correspon
      Estableix_paràmetre(2+5*i, llista_IDs[i])
      %Muntem la seqüència de bytes, cada paràmetre és un byte i cada
paraula es de 2
      Estableix_paràmetre(2+5*i+1, calcula_byte_baix(llista_posicions[i]))
      Estableix_paràmetre(2+5*i+2, calcula_byte_alt(llista_posicions[i]))
      Estableix_paràmetre(2+5*i+3, calcula_byte_baix(llista_velocitats[i]))
      Estableix_paràmetre(2+5*i+4, calcula_byte_alt(llista_velocitats[i]))
Fi Per

%La longitud total
Estableix_longitud_paquet((4+1)*Nombre_actuadors+4) %dxl_set_txpacket_length
%Enviem el paquet
Envia_paquet %dxl_txrx_packet

retorna Estat_comunicació %No hi ha paquet de retorn

Fi Funció
```

Finalment però, el ritme d'escriptura als servomotors va ser prou ràpid com per reescriure la posició al control de cada actuador abans d'haver executat el moviment sencer de la instrucció anterior. D'aquesta manera, quedava molt minimitzat l'efecte possible de tenir actuadors que en previsió d'un desplaçament molt gran descompensessin la uniformitat del moviment. És per això que el programa final no incorpora aquesta última rutina.

4.4.2. Problemàtica sorgida

La casa fabricant proporciona al públic diverses llibreries per a diferents entorns de programació. Pel que fa al software que s'empra en aquest projecte es va fer servir el mètode de crida de llibreria externa de Matlab®. Aquest mètode permet treballar amb llibreries de C o C# com la de Dynamixel™. Per tant, en un inici es va creure que es podria realitzar amb facilitat la comunió entre un script o un programa Matlab® que construís els paquets més complexos i realitzés les operacions desitjades, amb el programa de Simulink®.

La problemàtica que sorgí fou la següent. Simulink®, abans d'iniciar el seu cicle d'operació a través d'iteracions, i per tal d'executar els blocs a una velocitat adequada, compila el sistema programat. Però quan s'implementa un bloc amb codi de Matlab® a l'interior, el software no permet ni compilar crides de llibreries en un llenguatge diferent, ni l'ús de classes (orientat a objectes) de Matlab®. El compilador necessita un codi de funcions bàsiques. Aquest fet va comportar un canvi substancial a la manera d'enfocar la programació de les rutines desitjades.

La necessitat de disposar de blocs de Simulink® que treballessin amb una llibreria externa va suposar l'ús de l'eina *Legacy Code*. Es va haver d'adaptar tot el codi al format acceptat per aquesta eina, explicada a continuació.

4.4.3. Legacy Code

En general, el terme *Legacy Code* fa referència a fer servir codi en un llenguatge no suportat en el software que es pretén utilitzar. L'eina que proporciona Matlab® de *legacy code* permet crear blocs de Simulink® que executin funcions de llibreries externes en C i era la solució necessària per seguir endavant. Aquesta metodologia però, feia necessari ampliar la llibreria original de Dynamixel™ SDK i incorporar-hi les rutines programades prèviament en Matlab®.

Arxius necessaris

Matlab® necessita tenir al seu *path* tot un conjunt de fitxers amb dades referents a les llibreries i extensions ha de fer servir i tenir carregades. En el cas de llibreries en llenguatge C o C# usa arxius dels tipus:

- Lib.lib → Arxiu que vincula diferents elements d'una llibreria dinàmica i n'organitza els submòduls.
- Lib.dll → Totes les dades necessàries per l'ús de la llibreria.
- Lib.def → Arxiu hexadecimal amb més informació sobre la llibreria compilada.
- Lib.c →Arxiu en C amb el codi.
- Lib.h → Arxiu de *headers* on es troben els noms de totes les funcions d'aquesta llibreria.

Aquests arxius estan molt vinculats entre sí ja que són part de l'estructura de qualsevol programa o llibreria en codi C. Quan les llibreries es compilen (el codi C es llenguatge compilat) es generen alguns d'aquests fitxers, mentre que d'altres són part de la fase de desenvolupament i programació.

Tant per fer servir les funcions necessàries dins del Simulink® com per fer servir directament funcions implementades i afegides a la llibreria des del Matlab®, era necessari obrir els fitxers de la llibreria Dynamixel™ SDK i extendre-la en un format semblant en el que va ser programada. Posteriorment, és necessari estructurar els fitxers i compilar de nou tota la llibreria per tal de poder-la usar normalment a Matlab® com s'ha descrit inicialment. La part interpretable d'aquests arxius modificats amb les funcions implementades noves es l'extensió del codi de la llibreria dels *Software Development Kit* de l'annex, citada anteriorment.

Les funcions en *legacy code*

Una vegada es va haver obtingut tots els fitxers necessaris per treballar en Matlab® i per emprar el *legacy code* es va procedir de la manera següent. El procés descrit a continuació es realitzà mitjançant un script de Matlab® amb les diferents instruccions.

Les funcions implementades es van anomenar:

- Escriptura síncrona: "dxl_sync_write_word"
- Escriptura de velocitat i posició del moviment : "dxl_set_goal_speed"

Inicialment, es defineix un tipus de dades del tipus *legacy code* estructurades amb les propietats següents:

```
def =

               SFunctionName: ''
InitializeConditionsFcnSpec: ''
              OutputFcnSpec: ''
               StartFcnSpec: ''
          TerminateFcnSpec: ''
               HeaderFiles: {}
               SourceFiles: {}
              HostLibFiles: {}
            TargetLibFiles: {}
                  IncPaths: {}
                  SrcPaths: {}
                  LibPaths: {}
                SampleTime: 'inherited'
                   Options: [1x1 struct]
```

Tot aquest conjunt de propietats són les que es poden fer servir per definir un bloc de Simulink®. No totes les propietats són necessàries en aquest cas ja que hi ha diferents tipus de blocs amb diferents funcionalitats. Amb aquesta estructura es prepara la informació que necessita el programa per crear el bloc.

És important destacar que la simulació en Simulink® presenta 3 fases de funcionament:

1. Inicialització: Carrega funcions i dades d'inici de la simulació.

2. Funcionament: Correspon al comportament del bloc durant la simulació.

3. Finalització: Processos i funcions d'acabament de la simulació.

En aquest cas doncs, fou necessari un bloc que permetés, a cada iteració, escriure noves instruccions als actuadors, dictades pel control. Aquest bloc havia de ser capaç d'obrir la comunicació inicialment, escriure les instruccions que rebés, i finalment tancar la comunicació. Una funció diferent per cada etapa descrita.

Procediment

Per fer això, cal assignar a les propietats de una variable de *legacy code* amb unes característiques i una sintaxi concreta. Per definir amb precisió cada funció de C cal indicar a Matlab® l'equivalència entre tipus de dades en llenguatge C i Matlab®, així com si ha de treballar amb paràmetres ('p') i/o variables d'entrada ('u'), i indicar les variables de sortida si en té ('y').

La funció emprada d'escriptura als actuadors síncrona -anàloga a l'altra funció programada- es va definir de la manera següent:

Arranca el programari:

```
sync_write_word = legacy_code('initialize')
```

Es defineix una funció d'inici de la simulació, en aquest cas obertura del port i inici de la comunicació:

```
sync_write_word.StartFcnSpec = 'dxl_initialize(int32 p1,int32 p2)';
```

Es defineix la funció que tindrà lloc durant la iteració, en aquest cas l'escriptura als actuadors:

```
sync_write_word.OutputFcnSpec='int32 y1 = dxl_sync_write_word(int32
p1, int32 p2, int32 p3[], int32 u1[])';
```

Es defineix una funció d'acabament d'iteració, en aquest cas s'acaba la comunicació:

```
sync_write_word.TerminateFcnSpec = 'dxl_terminate()';
```

Es pot donar al programa fitxers *.c i *.h per a què els compili, però la tasca ja estava feta ja que les proves que es van realitzar sobre les funcions programades en llenguatge Matlab® ja

requerien d'arxius compilats. Només cal indicar el nom dels arxius de la llibreria d'on obté les funcions.

```
sync_write_word.HostLibFiles = {'dynamixel.lib'};
```

L'arxiu amb les capçaleres de les funcions:

```
sync_write_word.HeaderFiles = {'dynamixel.h'};
```

S'indica el nom del bloc:

```
sync_write_word.SFunctionName = 'sync_write_word';
```

Es genera el bloc:

```
legacy_code('generate_for_sim',sync_write_word);
```

```
legacy_code('slblock_generate',sync_write_word);
```

Aquest bloc generat té l'aspecte observable a la figura 4.3. Es pot veure que té una entrada per rebre la llista o vector de noves posicions pels motors i una sortida per retornar l'estat de la comunicació.

Pel que fa als paràmetres, aquests són visibles i editables en la finestra d'opcions del bloc i també les funcions que inclou i la relació entre ells.

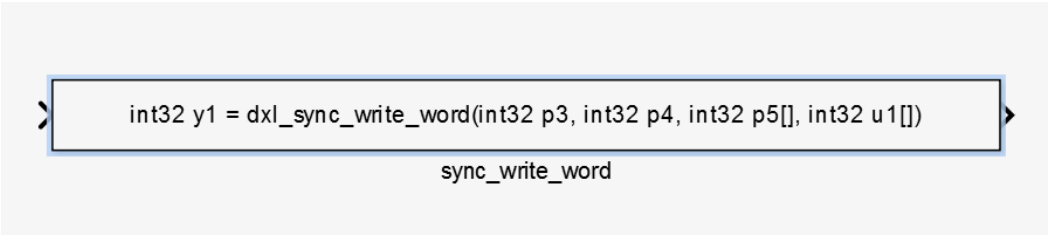


Fig 4.3. *Resultat de la creació de blocs mitjançant el* legacy code.

En aquest cas, P1 és el port de comunicació, P2 el *baud*, P3 el nombre d'actuadors, P4 l'adreça on començar a escriure i finalment P5 és la llista d'IDs tot seguint l'indicat prèviament en l'apartat 4.4.1. Es pot veure com aquesta nomenclatura concorda amb la definició de les funcions introduïda durant la generació del bloc (veure figura 4.4).

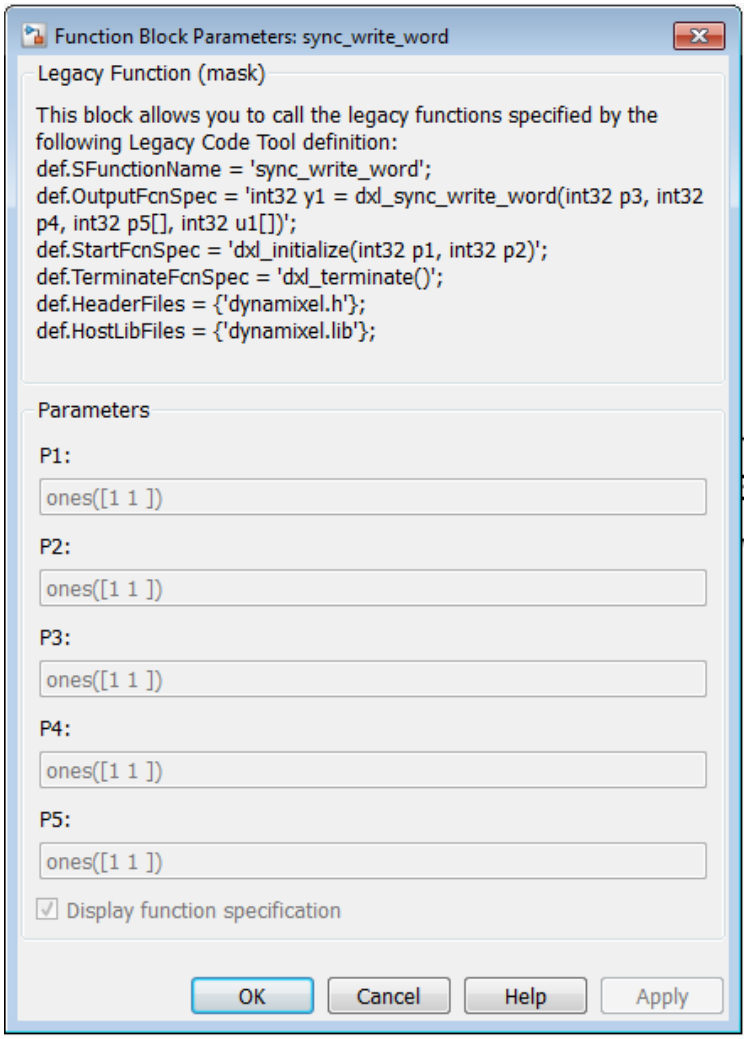


Fig 4.4. *Menú desplegable del bloc creat amb tota la informació que el defineix.*

5. Visió

Una vegada la comunicació amb els actuadors va estar garantida i funcional, aquests havien de ser capaços de reaccionar davant d'una informació provinent dels sensors. En aquest apartat es tractarà com la càmera introdueix informació al llaç de control. En primer lloc s'explicarà el processat de les imatges preses i posteriorment el tractament d'aquesta informació.

5.1. Imatges digitals

El primer que cal definir quan es parla del processat de les imatges és la manera com es digitalitza aquesta informació.

5.1.1. Espais de color

A la pràctica, qualsevol color es representa mitjançant el que es coneix com a coordenades de color. Això vol dir que, fent servir una tupla de 3 dimensions que és el més comú, es representa un color visible per l'ull humà. Cada valor de la tupla representa el valor de la dimensió en qüestió que es vulgui tractar i que pot ser de diferents natures: saturació, lluminositat, intensitat de color... És d'aquesta manera com es situen els diferents color d'un espai que finalment és discret ja que hi ha un nombre limitat de colors indexats en el sistema informàtic que el representa i, per tant, és un rang finit de color al qual es pot accedir.

A continuació es detallen els espais de color més comuns:

RGB

De l'anglès *Red, Green and Blue*, és el sistema de composició de color a partir de les intensitats dels colors primaris amb què es forma. És un model de color que es basa en la idea de la síntesi additiva, el que vol dir que és possible la creació d'un color a partir de la suma de tres colors llum primaris.

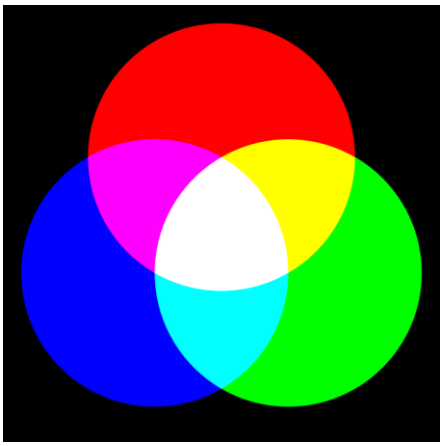


Fig 5.1 *Composició de colors en RGB.*

HSV

De l'anglès *Hue, Stauration and Value*, defineix un model de color en termes dels seus components constituents en coordenades cilíndriques.

La tonalitat correspon al tipus de color i es representa com un angle en un cercle de colors que va de 0 a 360º tot i que sovint també es normalitza de 0 al 100%.

La saturació es representa com la distància a l'eix de brillantor negre-blanca i els valors sovint es defineixen també del 0 al 100%. A aquest valor també se l'anomena puresa del color.

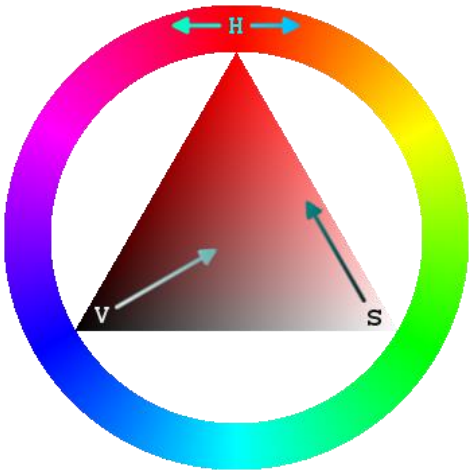


Fig 5.2. *Representació de l'espai de colors HSV [19]*

Finalment, el valor, que correspon a la brillantor del color. Estableix la quantitat de negre que se li afegeix al color de llum escollit per la tonalitat. El valor 0 sempre causa un color 0. El 100% correspon a la tonalitat triada, amb la saturació adequada sense cap negre afegit.

YCbCR

Es tracta d'una base colorimètrica que és emprada sovint en sistemes de transmissió de vídeo i de televisió. Es fa servir degut al fet que la base de colors primaris RGB conté molta informació redundant i aquest fet està vinculat amb l'eficiència de codificació. Això vol dir que sovint caldrà menys capacitat per transmetre imatges en format digital emprant aquesta altra codificació.

La luminància o "Y" correspon a una suma ponderada de les components RGB, que sovint és el format original en què es capta la informació. Aquest paràmetre és el més important ja que conté la informació més rellevant de la imatge. El sistema visual humà és molt més sensible a aquesta informació que als colors.

"Cb" és la component de color blau de la imatge.

"Cr" és la component vermella de la imatge".

L*a*b*

Aquest espai de color fa servir un paràmetre d'intensitat ("L") i dos paràmetres de crominància ("a" i "b") que descriuen el color. Ha estat especialment estudiat perquè les distàncies entre colors corresponguin a les diferències percebudes per l'ull humà.

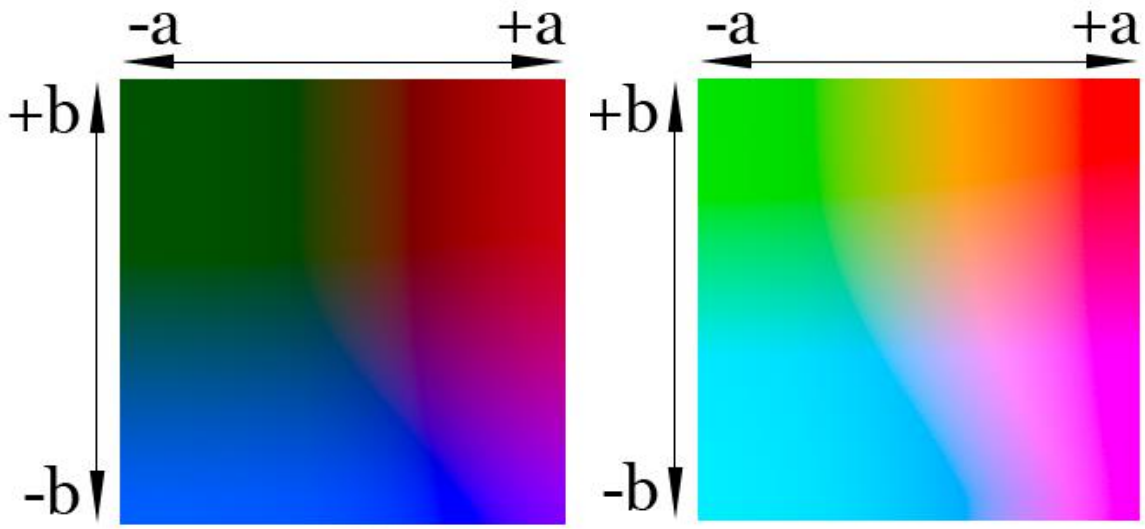


Fig 5.3. *Representació de l'espai de colors L*a*b* per a diferents lluminositats "L" [20]*

5.1.2. Matrius d'imatge

En format digital doncs, les imatges són matrius de píxels N x M on N i M corresponen a la resolució de la imatge. Com s'ha explicat amb anterioritat, ja que cada píxel conté un color definit amb 3 coordenades, obtenim 3 matrius per cada imatge en color. Cada matriu representa els valors de cada canal per tots els píxels d'aquesta.

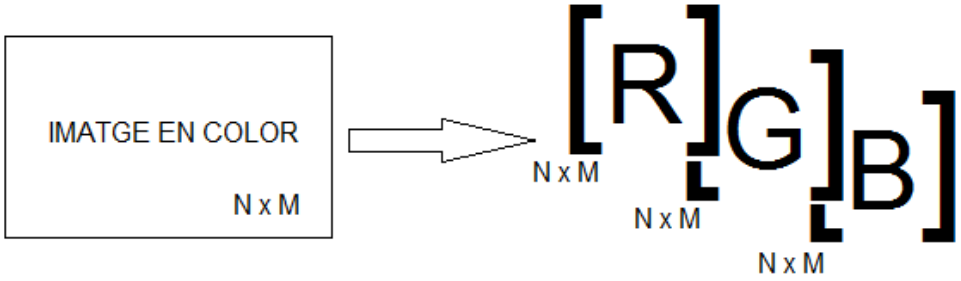


Fig 5.4. *Representació de la descomposició en matrius d'imatge per cada canal*

5.2. Muntatge

El muntatge del sistema de visió és el que es pot observar en la figura següent:

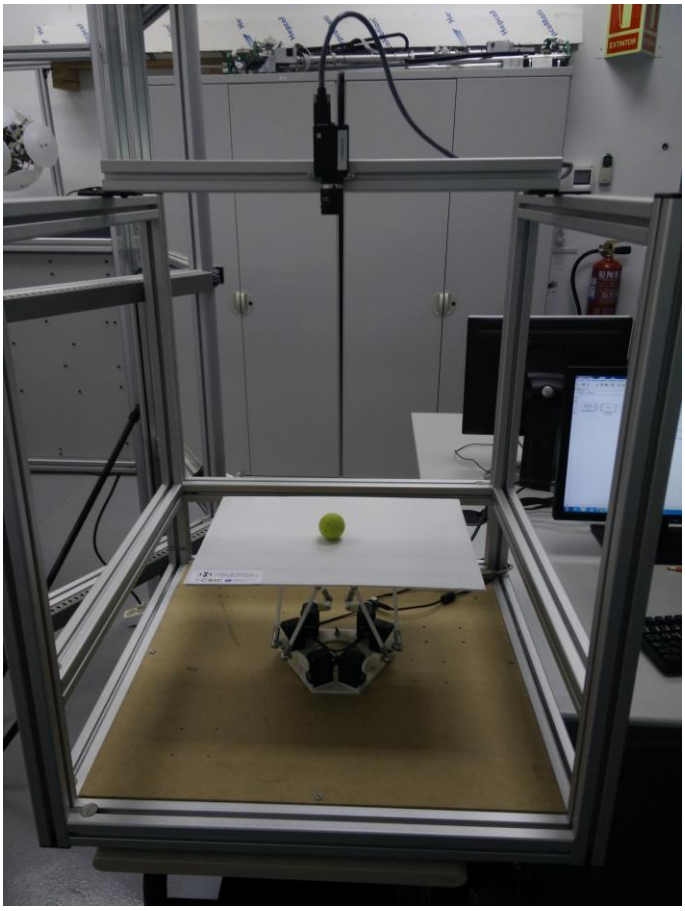


Fig 5.5. *Fotografia del muntatge final de robot i cámara zenital*

El sistema està muntat en una estructura de base quadrada 68x68 cm i 75 cm d'altura. La càmera es troba a la part superior de l'estructura. Tant la base de la plataforma com la càmera es troben fixades a l'estructura per mantenir constant la orientació relativa entre ambdues, ja que qualsevol canvi influiria al processat de les imatges, que fa servir aquesta informació.

La orientació i distància de la càmera permeten que per a qualsevol configuració de la plataforma, mentre la pilota es trobi sobre d'aquesta, sempre pertanyerà al camp de visió de la càmera.

El rang de la càmera

La imatge típica obtinguda per la posició de la plataforma en repòs és la següent:

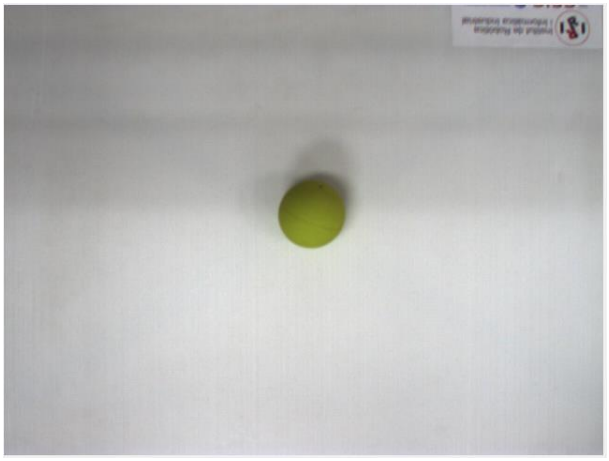


Fig 5.6. *Fotografia presa des de la càmera en el muntatge definitiu.*

La plataforma ha estat dimensionada per caure ajustada dins del rang de visió de la càmera per aquest muntatge. Cal esmentar que posteriorment s'ha hagut de jugar amb les resolucions que oferien un rendiment millor i que aquestes, sovint, modifiquen lleugerament l'angle d'obertura representat en les imatges finals, ja que no es prenen tots els píxels que ofereix el sensor.

El centre de la plataforma està centrat en la imatge de la càmera i els eixos X i Y de la càmera són paral·lels a les vores de la plataforma en repòs. La normal de la càmera i la normal de la plataforma s'ha procurat que siguin colinials per aquesta configuració o el que és el mateix, paral·lels entre si pla de la imatge i pla de la plataforma.

Posició i aspectes rellevants de la càmera i la plataforma

La altura del centre de la plataforma en repòs es troba a 22 cm. L'objectiu de la càmera es troba a 69 cm d'altura. Això deixa 47 cm de la superfície a la càmera.

Els angles d'apertura en X i en Y són de 40º i 36º respectivament per la resolució triada finalment i justificada en l'apartat 4.2. La plataforma té unes dimensions de 40x30 cm.

La pilota i la superfície han canviat durant el treball degut a les diferents respostes dinàmiques ofertes. La pilota final és la present en les imatges i té 2 cm de radi.

5.3. Processat i anàlisi de les imatges

El procés descrit en aquest apartat sobre la binarització ha tingut un caràcter iteratiu i ha estat modificat i repetit en múltiples ocasions degut a canvis en color textura i mida dels elements durant l'evolució del projecte.

5.3.1. Binarització

Pel cas que ens ocupa, la visió ha de ser capaç de determinar la posició relativa d'una bola o pilota en l'espai en relació amb la càmera. En no aportar informació necessària o rellevant la seva orientació i dada la simetria esfèrica que fa que qualsevol secció (o orientació i posició en la imatge) correspongui a un cercle, es pot fer servir el seu centre de massa per determinar la posició. Les coordenades d'aquest centre de masses en la imatge, suposant la idealització que l'esfera és isòtropa, correspondrà a les coordenades del centroide de la zona binaritzada.

Ja s'ha comentat que s'han de minimitzar els recursos emprats en cada iteració i aquest processat d'imatges tendeix a ser molt costós en aquest aspecte. És per això que es decideix fer només una binarització de la pilota mantenint fixa la orientació de la càmera respecte la base del robot així garantint que amb la informació del centroide en la imatge i les posicions de càmera i robot n'hi hagi prou per controlar la posició de la bola.

Una imatge de format digital amb 3 canals amb valors de cada element de les matrius acotats pel nombre de bytes que defineixen cada píxel es converteix en una sola matriu de valors booleans o binaris (0 ó 1).

Aquesta conversió es duu a terme mitjançant diferents algorismes segons l'objectiu de la binarització. Aquests algorismes actuen en les matrius d'imatge recorrent els seus elements per tal d'extreure la informació desitjada. Sovint es busquen colors, formes, canvis bruscos en lluminositat, etc. Per tal d'obtenir una imatge binaria doncs, cal realitzar en algun punt del processat comparacions de valors o la busca d'igualtats, que aportin aquest caràcter booleà a la imatge. A cada píxel o conjunt de píxels se li aplica l'algorisme que finalment, comparat amb un valor llindar triat (*threshold*) en resulta un valor 0 ó 1.

És convenient que, en previsió de possibles "intrusos" en la imatge (una mà, una eina, el contorn de la plataforma en rotar fortament, reflexos,...), el sistema sigui robust i es centri en el color de la pilota amb prou precisió. El fet que la càmera no suporti la emissió de la imatge en tots els espais de color definits caldrà tenir-lo en compte per agilitzar el processat.

Per aprofitar tot el possible les eines a disposició, es va recórrer a l'ús d'una aplicació d'extracció d'objectes d'imatges basada en el color anomenada *Color Thresholder*. Fent servir aquesta aplicació, es pren una imatge (amb la càmera ja muntada, enfocada i ajustats els paràmetres òptics) de la pilota desitjada en la superfície de control. Aquesta aplicació desplega un menú amb la imatge oberta en diferents espais de color per tal de donar una orientació sobre quin pot ser el més adient per treballar. Es representa cada canal en una imatge en blanc i negre, on el que queda representat és la intensitat del valor del canal en cada píxel en un rang d'escala de grisos.

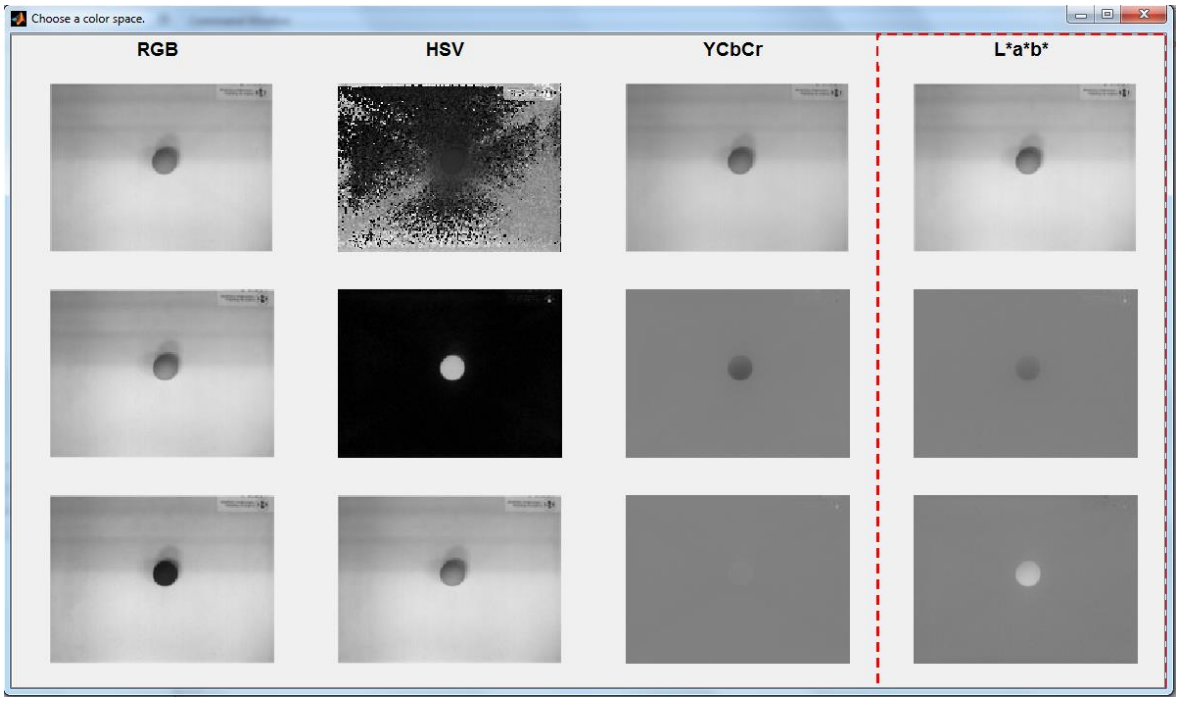


Fig 5.7. *Captura de pantalla de la representació en intensitats de cada canal en 4 espais de color.*

Una vegada escollit l'espai de color, en aquest cas l'espai escollit és el L*a*b* degut a la claredat apreciable d'un color com el verd clar en un fons blanc-groguenc en el canal b*, es procedeix a jugar amb els valors dels llindars per cada canal. La decisió de prendre un espai de color i no un altre també arrossega l'experiència del rendiment observat en cada prova, així com la robustesa que aporten i la velocitat del processat.

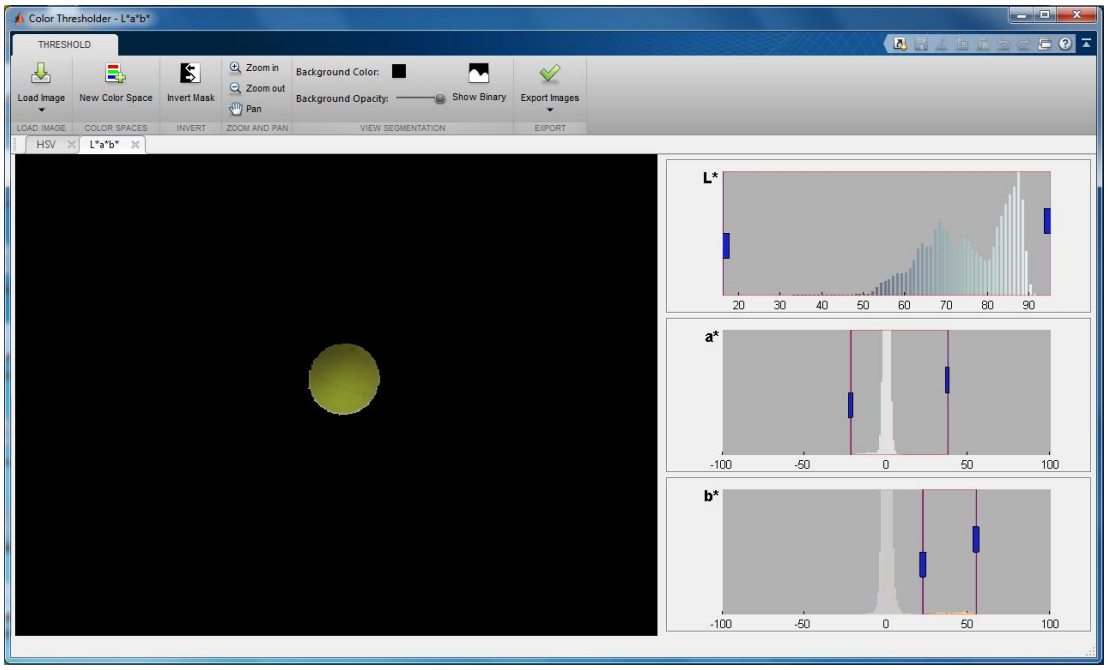


Fig 5.8. *Captura de pantalla de la l'extracció del color d'interès en espai L*a*b*.*

L'ideal és haver de fer servir només un llindar en un canal per tal de no haver de recórrer les 3 matrius. Tal com s'havia previst, només ajustant els rangs de valors del canal b* a un valor per sobre de 20, s'aïlla completament la pilota de la resta de la imatge.

5.3.2. Esquema de visió de Simulink®

De la mateixa manera que l'apartat 5.3, aquest també ha presentat una evolució constant, ja que s'han realitzat tota mena de proves per tal de garantir la fluïdesa de la resposta. Aquesta fluïdesa ha estat apreciable en els fotogrames per segon de la imatge binaritzada. És per això que hi ha un *display* de vídeo a la sortida. A través del representat en aquesta finestra de vídeo, es podia apreciar que de la seva fluïdesa en depenia la fluïdesa de moviments del robot. En les proves dutes a terme es va buscar constantment evitar que el coll d'ampolla de tot el control fos la visió i va condicionar constantment el mètode d'obtenció del centroide.

L'esquema blocs

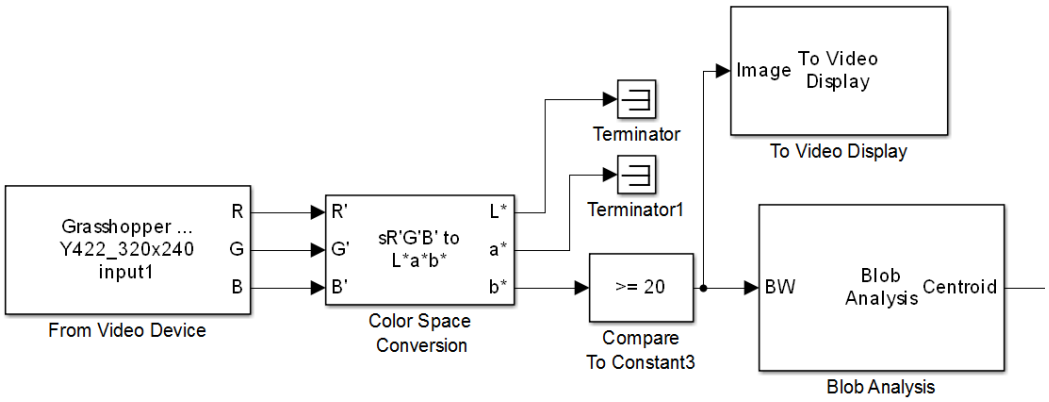


Fig 5.9. *Esquema del diagrama de blocs obtingut de la visió al Simulink®*

En la figura 5.9. es pot veure l'organització del sistema de visió al Simulink®. Aquest està format pels blocs següents:

From Video Device: Aquest bloc pren les imatges des d'un dispositiu de vídeo. En aquest cas es pot veure que reconeix el perifèric Grasshopper. N'indica la resolució i els canals de color en què emet. La màxima resolució que no alenteix el sistema fins a fer-lo incontrolable és la de 320x240 píxels (ben bé el doble en ambdós eixos del que era possible amb una web cam convencional). Per a aquesta resolució només s'emet un senyal en YCbCr a un màxim de 60 fotogrames per segon, però la mateixa càmera ho passa a RGB emprant el seu propi convertidor. Aquesta conversió és necessària degut a que el bloc que ve a continuació no permet la conversió directa de YCbCr a L*a*b*, i cal aprofitar al màxim els recursos disponibles.

Color Space Conversion: Aquest bloc passa d'un espai de color a un altre aplicant l'algorisme adequat a cada conversió. Permet les conversions següents:

- RGB a YCbCr i viceversa
- RGB a blanc i negre
- RGB a HSV i viceversa
- RGB a L*a*b* i viceversa
- RGB a XYZ i viceversa

Terminator: Aquest bloc simplement tanca senyals que no van a cap altre bloc.

Compare to constant: Compara un valor d'entrada amb una constant i en treu un valor booleà. És aquí on es comprova que el píxel en qüestió passi del llindar que li atorga un color com el del rang de la pilota. Cal recordar que aquest esquema és realitza *element wise,* el que vol dir que se sotmet a tot aquest tractament cadascun dels elements de les matrius d'imatge.

To Video Display: Permet una visualització del resultat de la binarització com es mostra a la figura 5.10.

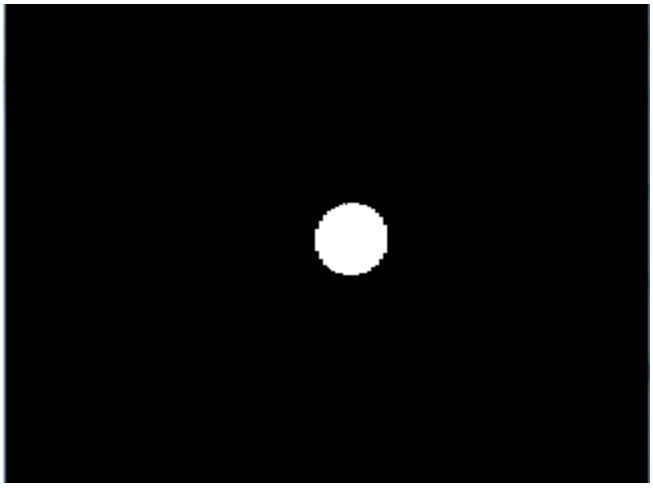


Fig 5.10*. Fotograma de la sortida de vídeo de la imatge binaritzada.*

Blob Analysis: Aquest bloc, finalment, és un bloc per defecte de la llibreria de Simulink® d'anàlisi d'imatges. Analitza de diferents maneres les regions disconnexes d'una imatge binaritzada. La connectivitat consisteix en el nombre mínim de vèrtex de píxels de mateix valor binari que estan en contacte per determinar si són pertanyen a la mateixa regió. Si és 2 correspon a una cantonada, 4 a una aresta i 8 (el que es fa servir al programa) implica que el píxel ha d'estar envoltat per 3 cares d'altres píxels de les mateixes característiques. La connectivitat es fa servir per descartar valors puntuals o regions binàries no contínues. Els paràmetres escollits es poden veure en les figures 5.11. i 5.12.

La manera més rudimentària de realitzar el càlcul seria recorrent els elements de la matriu binària i calcular la coordenada mitjana en X i en Y amb presència d'uns. Partint d'aquest concepte bàsic, també es pot realitzar una cerca intel·ligent basant-se en posicions prèvies de la pilota o amb càlcul paral·lel per zones de la imatge... Implementar aquests mètodes no

empra al màxim els recursos del computador i qualsevol bloc o rutina per defecte serà molt més eficient a l'hora de lidiar amb aquests problemes i més còmode de fer servir. A més a més, la binarització és senzilla i el programa permet trobar directament el necessari: el centroide.

El bloc dóna a la seva sortida les coordenades del centroide en la imatge (amb decimals).

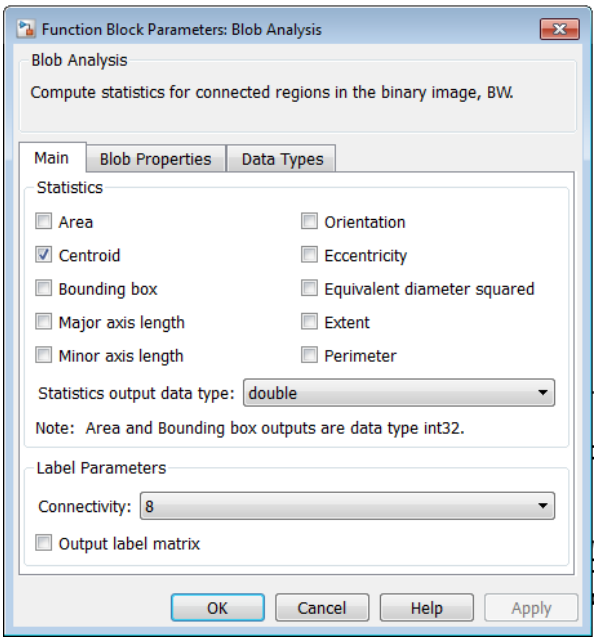


Fig 5.11. *Seleccions efectuades dins del menú principal del bloc* Blob Analysis.

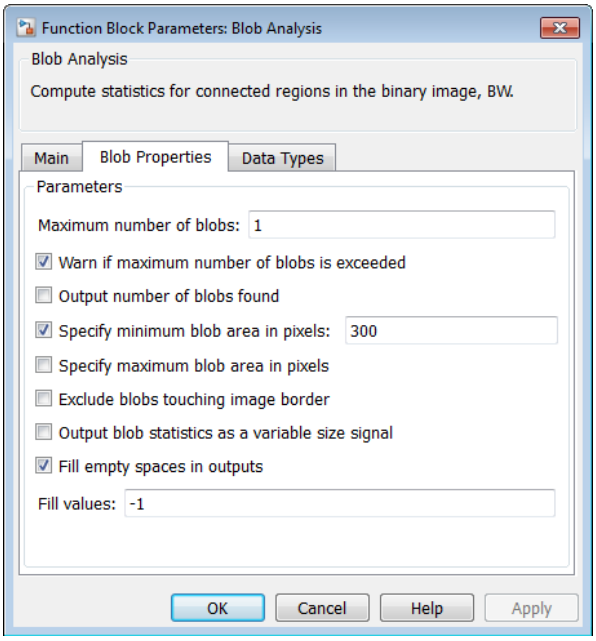


Fig 5.12. *Seleccions efectuades i paràmetres escollits dins del menú de propietats de la zona del bloc* Blob Analysis.

5.4. Tractament de les dades

Una vegada es varen tenir les coordenades en la imatge de la pilota es va voler donar un sentit físic a aquestes, intentant determinar la posició real en l'espai de la pilota. Aquest càlcul pretenia proporcionar al llaç de control valors invariants davant la resolució triada. D'aquesta manera s'assegura una mínima variació el comportament del sistema davant un canvi de resolució, les trajectòries programades en centímetres o metres per la pilota no es veuen afectades, etc.

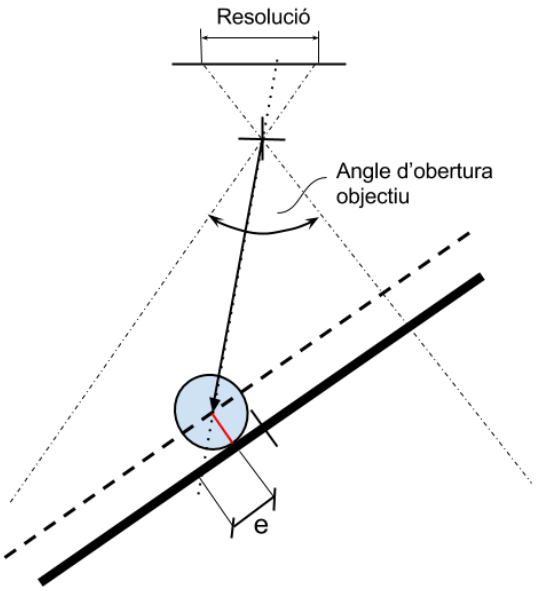


Fig 5.13. *Diagrama explicatiu de la problemática en l'avaluació de la posició.*

Amb l'ajuda del diagrama de la figura 5.13., s'observa una diferència entre la visió per la càmera i la posició real de la pilota en coordenades sobre la plataforma per una posició i orientació arbitràries. Degut a que la visió "veu" el centroide de la pilota, aquest correspon a la posició del centre de la pilota en la imatge però la imatge obtinguda és una projecció de la plataforma mòbil sobre el pla de la imatge o, equivalentment, sobre el pla de la plataforma mòbil en repòs.

Més endavant en l'avanç del projecte, es va poder veure que era indiferent a efectes pràctics el control bast-se en la posició projectada en un pla imaginari a l'altura del pla enfocat en la imatge (que presentava màxima coincidencia amb la realitat amb el de la plataforma en repòs)o bé controlar en coordenades de la plataforma. Tot i això, es va desenvolupar l'eina de càlcul per fer-ho possible.

Es va establir una relació entre les coordenades en píxels de la imatge i la recta en 3 dimensions que uneix el centre de projecció de la imatge amb el centre de la pilota per resoldre aquesta problemàtica.

5.4.1. Relacions entre els diferents sistemes de referència

En primer lloc, a partir de les coordenades de la imatge en píxels, que es defineixen des de la cantonada superior esquerra de la imatge, es centren al mig del pla, per fer coincidir amb el centre de la plataforma.

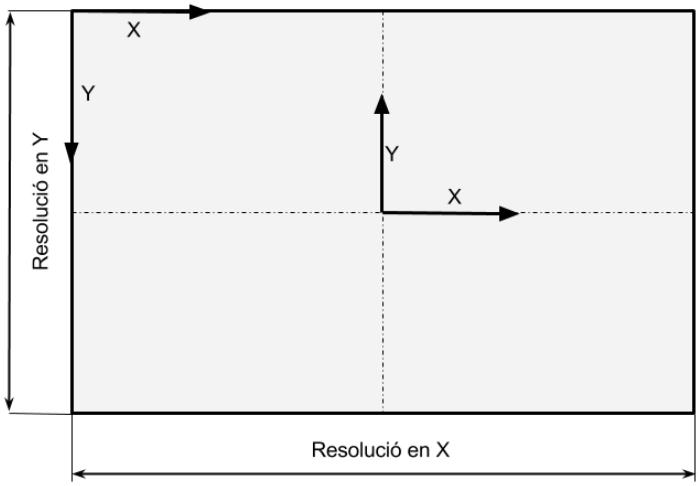


Fig 5.14. *Diagrama explicatiu de la primera transformació de les coordenades.*

Les coordenades d'un punt en X,Y de la imatge, doncs, corresponen a:

$$X_{corr} = \frac{RESOLUCIÓ_X}{2} - X;$$

$$Y_{corr} = Y - \frac{RESOLUCIÓ_Y}{2};$$

A continuació cal determinar els angles amb la vertical dels vectors V1 i V2 que corresponen a la descomposició del vector posició (vegeu figura 5.15.).

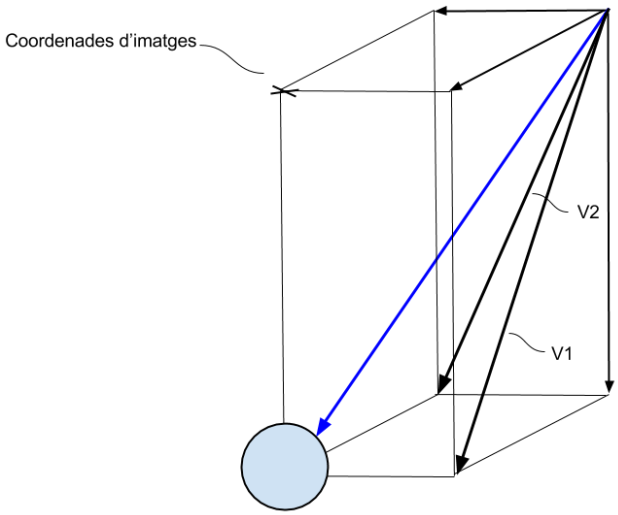


Fig 5.15. *Composició del vector que uneix l'objectiu amb la pilota*

La distància focal és la distància que hi ha des de l'objectiu fins al sensor on es projecta la imatge. Si aquesta es tingués, la relació entre la imatge enfocada i la realitat és molt més fàcil de trobar . En aquest cas però, només es coneix l'angle d'obertura de l'ojectiu en X i en Y de 42º i 36º respectivament. El procediment descrit a continuació és anàleg a cada coordenada.

L'angle amb la vertical de V1 si aquest correspon a X per exemple correspon a l'esquema representat a la figura 5.16.

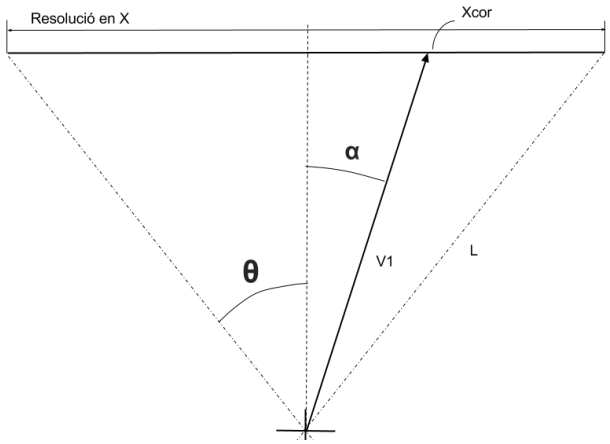


Fig 5.16. *Diagrama emprat per definir les relacions entre angles sense fer servir la distancia focal.*

I d'aquest, se'n poden deduir les relacions següents:

$$V_1 \sin\alpha = X_{cor};\quad V_1 \cos\alpha = L\cos\theta;\quad L\sin\theta = \frac{RESOLUCIÓ_X}{2}$$

I no és difícil demostrar que se'n pot extreure una igualtat per a α no depenent de cap distància focal:

$$\tan\alpha = \frac{X_{cor}}{RESOLUCIÓ_X}\tan\theta$$

Finalment, per definir el vector unitari en 3 dimensions que apunta des de l'objectiu fins la pilota com el definit en la figura 5.15., es calcula el següent:

S'imposa una component Z = -1. D'aquesta manera, amb les tangents corresponents a cada coordenada, es poden assignar directament com a components X i Y respectivament i respectant les proporcions del vector.

Posteriorment es normalitza aquest vector dividint-lo pel seu mòdul i s'obté el vector director de la recta amb la forma $v = [v_x, v_y, v_z]$.

5.4.2. Intersecció de la recta i el pla

Per aquesta etapa, es pren l'orientació de la plataforma amb els paràmetres *ballangle* i *rotangle* tal i com es descriu a la cinemàtica del robot, a l'apartat 6.2 i 6.3.

Per no resultar repetitiva la definició de les transformades en la cinemàtica, en aquest apartat només es descriurà la metodologia de resolució de la intersecció, així com la rutina programada per resoldre-la.

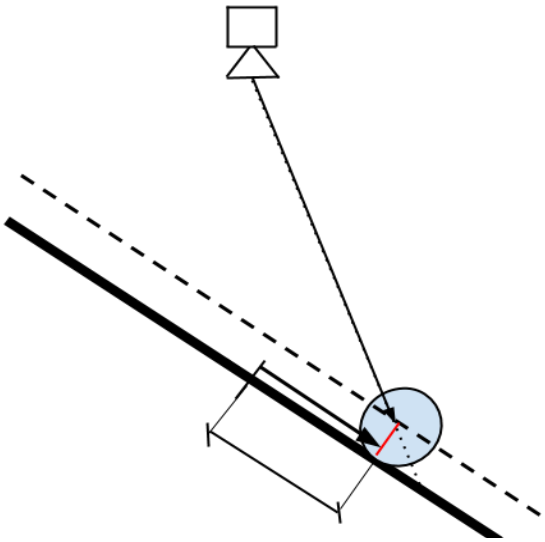


Fig 5.16. *Esquema representatiu de la intersecció calculada.*

Resolució

Aplicant les transformacions de la base al pla paral·lel a la plataforma ja rotada, es calcula el centre i el vector normal d'aquest. Un vector $[n_x, n_y, n_z]$ que permet expressar el pla en la referència absoluta com:

$$\Pi \equiv n_x x + n_y y + n_z z + d = 0$$

Aquesta expressió, avaluada en el centre del pla, permet trobar el terme independent *d*. La recta que passa pel centroide de la pilota com el punt de posició de l'objectiu en referència absoluta i un factor λ vegades el vector director definit anteriorment queda de la forma:

$$r = P_{càmera} + \lambda\,[v_x,\quad v_y,\quad v_z]$$

Es pot calcular la intersecció entre ambdues i traduir-la en coordenades sobre el pla. Fent servir les variables simbòliques *ballangle* i *rotangle* (veure l'apartat 6.3) s'ha calculat la solució amb Matlab® de la manera següent:

```
%Totes les mesures en centímetres
ALT_CAM = 47;
RADI = 2;
ALT_PLAT = 22;

%Es defineix la posició de la càmera en referència absoluta
POS_CAM = [0 0 ALT_CAM+ALT_PLAT];

%Es defineix el pla inclinat paral·lel a la plataforma a una distància R
%d'aquesta
RE = transl(0,0,ALT_PLAT);
TAXIS = transl(0,0,RADI)*trotz(ballangle + (pi/2));
TR =RE*TAXIS*trotx(rotangle);

%Es calcula el centre i el vector normal (fet unitari) d'aquest pla
centre = TR*[0;0;0;1];
normal = TR*[0;0;1;1];
normal = [normal(1);normal(2);normal(3)];
normal = normal/norm(normal);

%Es calcula el terme independent de l'equació del pla
tind = -[normal(1) normal(2) normal(3)]*[centre(1);centre(2);centre(3)];

%Es defineix la intersecció entre l'equació del pla i la recta que uneix
%centroide de la bola amb l'objectiu de la càmera
equation =  normal(1)*(POS_CAM(1) + lambda*vx)+ ...
            normal(2)*(POS_CAM(2) + lambda*vy)+ ...
            normal(3)*(POS_CAM(3) + lambda*vz) + tind;

%Es resol per lambda
lambda = solve(equation, lambda);

%Es calcula el punt d'intersecció en les seves 3 dimensinos
punt = [POS_CAM(1) + lambda*vx; POS_CAM(2) + lambda*vy; POS_CAM(3) +
lambda*vz; 1];

%Es passa de coordenades absolutes en coordenades del pla
punt_pla = simplify((TR)\punt);
```

Una vegada executat el codi anterior, s'obtenen les expressions per les coordenades x i y del punt d'intersecció següents:

Coordenada X=

```
-(45*(cos(rotangle) + 24)*(vy*cos(ballangle) - vx*sin(ballangle)))/(24*vz
+ vz*cos(rotangle) + vx*cos(ballangle)*sin(rotangle) +
vy*sin(ballangle)*sin(rotangle))
```

Coordenada Y=

```
(45*(24*cos(rotangle) + 1)*(vx*cos(ballangle) +
vy*sin(ballangle)))/(24*vz + vz*cos(rotangle) +
vx*cos(ballangle)*sin(rotangle) + vy*sin(ballangle)*sin(rotangle))
```

L'avantatge fonamental d'obtenir les expressions per les coordenades del punt d'intersecció d'aquesta manera és que el resultat és directament interpretable com a codi en Matlab® amb la garantia de que no conté cap error de transcripció.

5.4.3. Implementació en Simulink®

La implementació del processat de les dades en Simulink® es va realitzar mitjançant un bloc *MATLAB Function* degut a la mida de les expressions resultats del càlcul, que feien poc aconsellable la seva implementació representació mitjançant blocs. Aquest bloc únic integra totes les operacions descrites en apartats anteriors sobre tractament de dades i es troba a l'apartat de visió com es representa a la figura 5.17., a continuació de la binarització. El codi contingut en aquest bloc es pot trobar a l'annex.

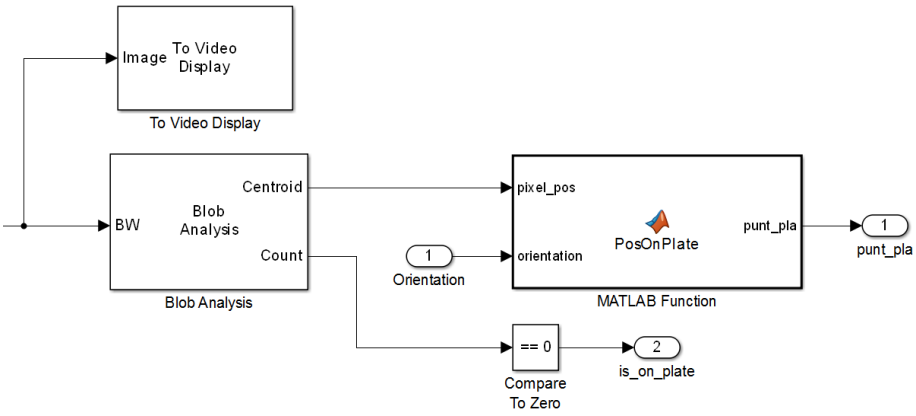


Fig 5.17. *Esquema del diagrama de blocs del post processat de la imatge*

Cal notar que rep una orientació de la plataforma pel port *orientation,* que necessita per calcular el punt en la plataforma, així com la posició en la imatge. Per la gran majoria de controls es va fer una estimació de la posició sobre el pla paral·lel a la base on la imatge estava enfocada.

El desenvolupament d'aquest sistema de visió corregit va ser una de les darreres modificacions realitzades. Al cap i a la fi, era indiferent pels objectius del projecte la diferència entre controlar sobre el pla de la imatge i controlar sobre la plataforma real.

6. El robot

Per realitzar la tasca de canviar la orientació i realitzar els moviments d'un pla per controlar la posició d'una pilota de seguida es va plantejar la possibilitat de fer servir un robot paral·lel. No hem d'oblidar que aquest treball s'ha realitzat a un laboratori de cinemàtica a on els robots d'aquest tipus, i en particular la plataforma Gough-Stewart, son l'objecte central d'estudi.

Així doncs, en aquest apartat es veurà en primer lloc què és una plataforma de Gough-Stewart i les seves característiques. A continuació, es descriurà l'arquitectura dissenyada per assolir els objectius d'aquest projecte, i es deduirà la seva cinemàtica inversa. Posteriorment, es tractarà la programació d'un simulador per tal de verificar que el model matemàtic es correcte. Finalment, es descriurà la plataforma construïda.

6.1. La plataforma de Gough-Stewart

6.1.1. Definicions

Una plataforma de Gough-Stewart, o simplement plataforma de Stewart, és un tipus de robot paral·lel que incorpora sis actuadors lineals que el doten de 6 graus de llibertat. Aquest tipus de robot té una versatilitat superior a les plataformes emprades en la bibliografia, ja que és capaç de rotar entorn qualsevol eix del seu espai de treball i permet per tant ajustar l'eix entorn el qual es vol rotar la plataforma per controlar la pilota. Amb aquest tipus de plataforma és possible, per tant, fer passar l'eix de rotació pel centre de la pilota.

Quan les cadenes cinemàtiques que uneixen la plataforma i la base són no prismàtiques, amb una distribució més o menys lliure, es parla d'un robot paral·lel generalitzat. La simetria en la distribució de les potes sovint garanteix simetries en l'espai accessible pel robot.

Un robot paral·lel es un sistema mecànic que fa servir múltiples cadenes cinemàtiques sèrie que connecten una plataforma mòbil a una base fixa. Quan tenim sis cadenes sèrie idèntiques amb un actuador cadascuna es considera, fent servir un abús del llenguatge, que es tracta d'una plataforma de Stewart. Aquest tipus de robots estan dins de la categoria dels robots paral·lels en oposició als robots sèrie per raons obvies.

6.1.2. Avantatges

Un manipulador paral·lel, normalment, està dissenyat amb cadenes cinemàtiques amb poques articulacions, simples, i que per tant presenti una rigidesa superior davant d'un moviment no desitjat en comparació amb un manipulador sèrie. Els errors de posicionament en cada cadena cinemàtica queden molt esmorteïts en la mitjana total de les altres cadenes mentre que, per altra banda, una cadena cinemàtica oberta presenta errors acumulatius en cada articulació i es veu reflectit en l'error de posicionament final. A més a més, donat que el pes de la plataforma mòbil es baixa en comparació amb la resta del robot i els motors es troben ancorats a la base fixa, la inèrcia de la plataforma en comparació amb la força provinent dels 6 actuadors és molt baixa i es sovint menyspreable en la dinàmica dels robots paral·lels.

Gràcies a això, el que en resulta és un robot molt més rígid, més precís, amb molta més capacitat de carrega, i més acceleració i velocitat que un sèrie de les mateixes dimensions. El preu a pagar és un espai de treball més petit.

Fig 6.1. *Algunes plataformes Stewart de l'Institut de Robòtica i Informàtica Industrial*

6.1.3. Arquitectura

El robot havia de complir amb una sèrie d'especificacions:

- Mida reduïda: El robot havia de ser capaç de moure boles de poc diàmetre, de la mida d'una pilota de ping-pong. Havia de ser transportable. Els actuadors AX-12+ son dels més petits de les sèries de Dynamixel™ presents al laboratori i no era necessari ni tenia sentit un robot d'una mida superior sobretot si el sistema es vol fer servir per jocs interactius de sobretaula.

- Velocitat d'actuació: Era d'importància que les cadenes cinemàtiques permetessin desplegar tot el potencial dels actuadors, ja que pel tipus de control necessari el punt clau del disseny es troba en les unions entre la plataforma i la base.

Actuadors prismàtics

És comú que els robots paral·lels estiguin actuats per actuadors prismàtics. Aquest disseny facilita la cinemàtica inversa, concretada a l'apartat següent, i aporta molta rigidesa a la construcció. Com que el que es disposa és de servomotors, els quals són molt més barats, cal transformar el parell i la rotació en un desplaçament i una força lineals. Si els servomotors ja inclouen una reducció elevada (254:1 pels AX-12+ i en aquest ordre per la resta), a més a més se'ls afegeix la reducció de la transmissió d'un vis-sens-fi, que es el més usual. Així s'augmenta encara més la rigidesa i la força del robot però aquest fet resulta en unes velocitats lineals extremadament baixes i no assumibles.

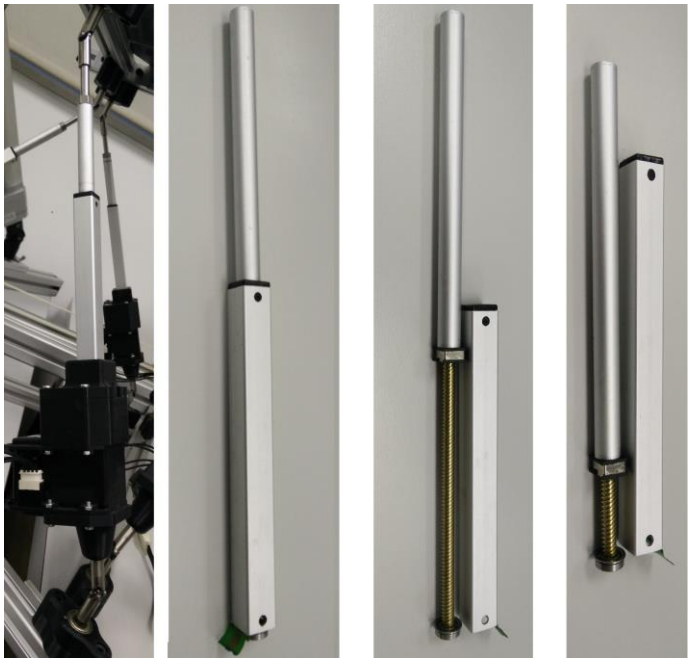


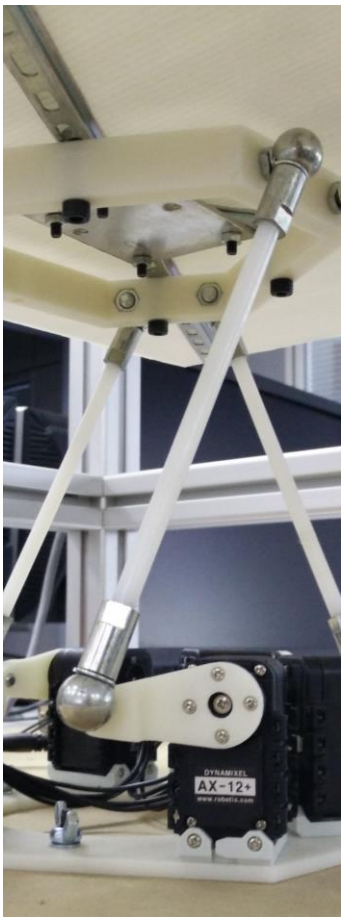
Fig 6.2. *Exemple d'una pota prismática on un vis sens fi s'enrosca entorn una femella per augmentar o disinuir a longitud de la pota.*

Tal com es pot apreciar a la figura 6.2., el motor es connecta a un vis sens fi que cargola una peça mòbil que, en no poder rotar, avança o retrocedeix lliscant per l'interior el tub prismàtic quadrat. El pas de rosca és de 5mm i afegeix encara més reducció, com ja s'ha comentat.

Amb el material del que es disposava al laboratori, el millor resultat obtingut fou una velocitat d'extensió de 0,5 cm/s amb els motors a 60rpm (velocitat màxima). Era evident que calia replantejar les cadenes cinemàtiques sèrie del robot.

Actuadors rotatoris amb mecanisme "biela-manovella"

Tot i que no es tracta del tot d'un sistema biela-manovella, ja que normalment fa referència a un mecanisme que converteix el moviment rotatori d'un eix en la translació rectilínia d'un altre, el principi és molt semblant. Es dota als actuadors d'un braç que empeny una pota rígida amb ròtules esfèriques a ambdues extrems, ja que cal seguir garantint els 6 graus de llibertat de la plataforma. És per això que el braç que actua com a manovella té un moviment en 3 dimensions enlloc de quedar comprès en un pla com en el mecanisme de biela-manovella clàssic (veure figura 6.3.).



Aquest sistema és capaç d'induir velocitats molt superiors en l'extrem lligat al braç del motor (amb un braç de 0,03m i les 60rpm s'aconsegueixen fins a 18,85cm/s de velocitat lineal a l'extrem del braç).

Les noves prestacions però, tenen un preu que correspon a la incorporació de singularitats en la cadena cinemàtica de cada actuador. Aquestes singularitats apareixen en el moment en que es troben alineats el braç de l'actuador i la pota en qüestió i en que apareixen dues possibles direccions de moviment i es topa amb una frontera de l'espai de treball.

Finalment, cal esmentar que aquesta opció també presenta una major acceleració i destresa en la configuració representada, ja que un mínim moviment del rotor del servomotor implica una component vertical de desplaçament de la pota més acusada i, en conseqüència, una variació en l'orientació de la plataforma més acusada.

Fig 6.3*. Detall de la cadena cinemática final.*

Disseny

En el disseny i muntatge del robot es varen emprar tot un conjunt d'eines de disseny assistit per ordinador i eines de taller del laboratori de cinemàtica de l'IRI. Es va procurar que el robot construït pogués complir amb les característiques prèviament definides.

Fig 6.4. *Fotografía del robot.*

Malauradament, degut a qüestions de confidencialitat d'alguns prototips de l'IRI que inclouen aspectes susceptibles de ser patentats en un futur, no es poden incloure aquí els models CAD ni els plànols del robot.

6.2. Cinemàtica del robot

La cinemàtica és la branca de la física que tracta els moviments de cossos sense tenir en compte les forces ni els parells involucrats. En el món de la robòtica es parteix de la idealització dels robots com a sistemes de cossos rígids connectats per articulacions. La cinemàtica de robots descriu l'orientació i posició (*pose*), velocitat, acceleració i totes les derivades d'ordre superiors de la *pose* dels cossos que conformen un robot.

Per altra banda, la dinàmica tracta les relacions entre les forces i parells presents en un sistema i les trajectòries, velocitats i acceleracions dels elements que el conformen.

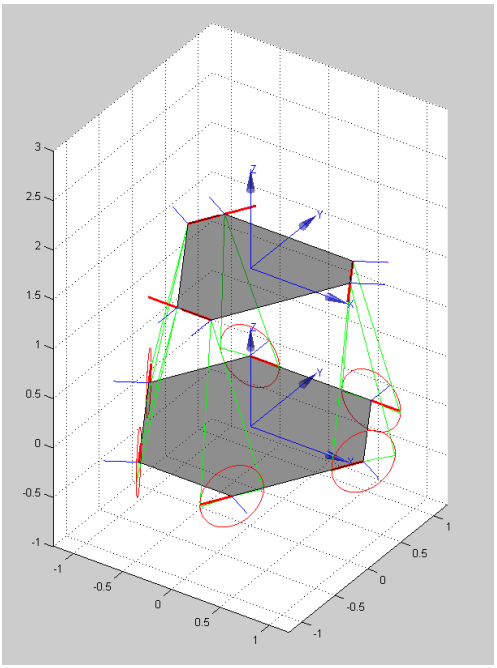


Fig 6.4. *Simulació del robot en repòs*

En un robot com el que és objecte d'estudi en aquest projecte, la cinemàtica pren tot el protagonisme, deixant la dinàmica en un segon pla. Aquest fet és degut a que, en el nostre robot, la força d'actuació en proporció a les forces d'inèrcia que es poden produir és molt gran. Amb unes característiques d'acceleració, velocitat i posicionament tan bones i només havent de determinar les orientacions desitjades en cada instant, el tractament de la cinemàtica era el que gaudia de tot l'interès. A més a més, en un sistema amb 6 graus de llibertat per l'actuador final, però amb 6 cadenes sèrie que treballen en 3 dimensions, tenen 5 graus de llibertat cadascuna, i estan conformades per 2 elements amb singularitats en el seu moviment, la complexitat de la dinàmica és important i pròpia d'un treball d'estudi exclusiu [21]. Els avantatges d'un estudi d'aquesta dinàmica es farien palesos per un control de la pilota sobre el pla enfocat de manera diferent a la tinguda en compte en aquest projecte.

Representació de la posició i la orientació

El mínim nombre de coordenades per descriure la *pose* d'un cos en un espai euclidià és de sis. Es defineix una referència *i,* consistent d'un origen expressat com a O_i, i una tríada de eixos de coordenades ortogonals expressada com a (x_i, y_i, z_i), fixos en un cos determinat. La *pose* d'aquest cos sempre serà expressada relativa a la d'un altre, així es podrà descriure com la *pose* d'una referència relativa a una altra.

- Posició d'un cos *i* relatiu a una referència *j:*

$$ {}^{j}O_i = \begin{pmatrix} {}^{j}O_i^x \\ {}^{j}O_i^y \\ {}^{j}O_i^z \end{pmatrix} $$

- Orientació d'un sistema de referència *i* relatiu al *j*, és una aplicació que transforma un vector expressat en base *i* a base *j*:

$$ {}^{j}R_i = \begin{pmatrix} \hat{x}_i \cdot \hat{x}_j & \hat{y}_i \cdot \hat{x}_j & \hat{z}_i \cdot \hat{x}_j \\ \hat{x}_i \cdot \hat{y}_j & \hat{y}_i \cdot \hat{y}_j & \hat{z}_i \cdot \hat{y}_j \\ \hat{x}_i \cdot \hat{z}_j & \hat{y}_i \cdot \hat{z}_j & \hat{z}_i \cdot \hat{z}_j \end{pmatrix} $$

La orientació es representa mitjançant matrius de rotació. La orientació de la referència *i* expressada en termes de la referència *j* es pot descriure expressant els vectors de la primera en termes de la segona.

També existeixen altres representacions, com ara els angles d'Euler (α, β, γ) que representen tres rotacions consecutives entorn als eixos d'una referència que va arrossegant a la resta mentre es mou i pels quals existeixen diferents combinacions (ZYZ, ZYX, ZXZ...). O bé els quaternions, que per simplificar es poden veure com a vectors unitaris de la forma: $q = (\cos\theta/2, w_x\sin\theta/2, w_y\sin\theta/2, w_z\sin\theta/2)^T$.

Per tal de representar una *pose* es fa servir una matriu 4x4 anomenada transformada en coordenades homogènies de la forma següent:

$$ {}^{j}T_i = \begin{pmatrix} {}^{j}R_i & {}^{j}p_i \\ 0_{1x3} & 1 \end{pmatrix} $$

A on la quarta columna representa la translació a aplicar al sistema de referència *i* per tal que el seu origen coincideixi amb el origen del *j*.

Cinemàtica directa i cinemàtica inversa

El problema de la cinemàtica directa consisteix en trobar la posició final de l'actuador o l'eina en funció de les translacions i orientacions dels actuadors ja siguin prismàtics o de rotació.

Per una cadena sèrie és el producte de les transformades que hi ha des d'una referència en què es vol expressar la *pose* de l'eina fins a ella. Aquesta operació es pot aplicar per descriure qualsevol relació entre punts si es tenen totes les transformades que els uneixen, ja sigui de manera simbòlica o física. Per exemple, les relacions per una transformada de la referència 0 a la 3 és del tipus:

$$^{0}T_{3} = {}^{0}T_{1} \cdot {}^{1}T_{2} \cdot {}^{2}T_{3}$$

Per un robot sèrie la solució de la seva cinemàtica directa és òbviament única ja que no conté cap articulació passiva, totes les articulacions són actuades. En canvi, per un robot paral·lel, la situació és molt més complicada. Una plataforma de Gought-Stewart genèrica pot arribar a tenir fins a 40 solucions per la seva cinemàtica directa. Introduint alineacions, simetries, coincidències, etc, es pot reduir aquest número fins a un mínim de 8 solucions per a un manipulador de tipus 3-2-1 [22] .

La plataforma de Gough-Stewart no genèrica més comuna és la anomenada octaèdrica que es mostra a la figura següent. Aquest tipus de plataforma pot arribar a tenir fins a 16 solucions per la seva cinemàtica directa.

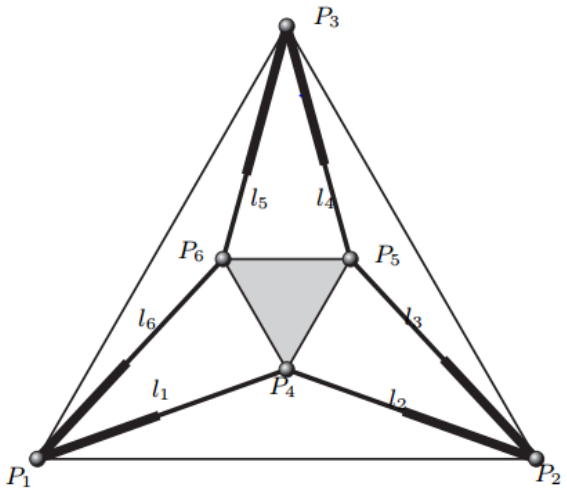


Fig 6.5. *Plataforma paral·lela octaèdrica.*

En una plataforma octaèdrica hi aparecixen articulacions esfèrique multiples. El disseny mecànic d'aquest tipus d'articulacions és complicat. Com a consequencia, s'han proposat unes modificacions per evitarles sense perdre part de les propietats de la plataforma octaèdrica. Alguns exemples d'aquestes modificacions es poden apreciar a la figura següent.

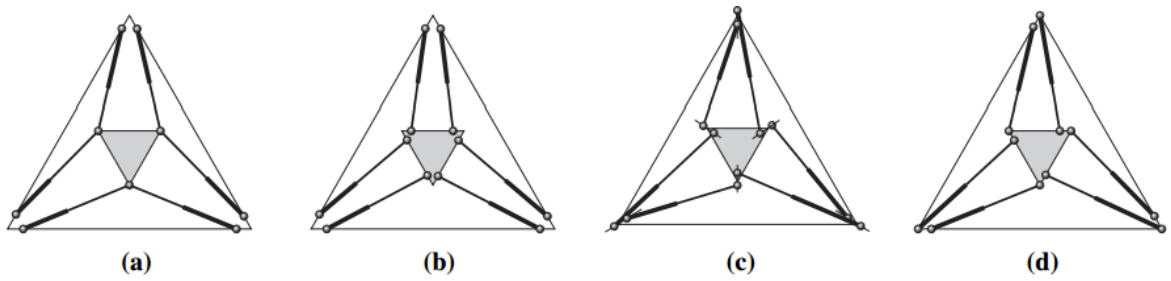


Fig 6.6. *Diferents solucions al muntatge de les ròtules amb mateix nombre de solucions a la cinemática inversa* [23]

El problema de la cinemàtica inversa, en canvi, consisteix en, partint d'una orientació i posició de la plataforma mòbil, determinar els valors dels actuadors del robot a través de les condicions de clausura en cada pota. Per resoldre aquest problema, es parteix de la locatlització dels extrems de cada pota i es resol la cinemàtica inversa per cadescuna d'elles per separat.

Ja s'ha exposat que, per tal de controlar la posició de la pilota, s'haurà de determinar els valors dels actuadors que fan posible assolit una determinada *pose* per la platarma móbil. S'haura doncs de resoldre la cinemàtica inversa del robot dissenyat.

Procediment

El procediment genèric, tot i que pot harver-hi altres maneres d'enfocar-lo, consisteix en definir una referència absoluta a la base del robot i una altra a la plataforma. Aquestes dues referències estan relacionades per una transformada homogènea que evidentment haurà de ser variable degut als graus de llibertat de la plataforma. Els punts definits respecte de cadascuna d'aquestes referències, i que serviran per trobar la solució, es poden definir com a simples vectors des de la referència a la que són fixos o, si s'escau, com els origens de nous sistemes de referència definits mitjançant transformacions constants que les relacioni amb la referència a la que són fixes.

Una vegada tots els elements amb distàncies invariants estan relacionats, es determina una configuració final per la referència de la plataforma i la transformada entre la base i aquesta. Amb tota aquesta informació, ja es poden determinar les equacions a resoldre..

6.3. Càlcul de solucions

En aquest apartat s'enfronta la cinemàtica inversa específica pel robot que es fa servir en aquest projecte. S'explica pas a pas la definció de referències del robot i l'obtenció de les solucions analítiques pels angles dels actuadors.

Abans però, cal definir alguns abreujaments per compactar la notació utilitzada:

- Per la translació pura entre dues referències unides per un vector en 3 dimensions:

$$Transl(\{x,y,z\}) = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Per rotacions pures d'angle θ entorn als eixos de la base:

$$TRotX(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sin\theta & 0 \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$TRotY(\theta) = \begin{pmatrix} cos\theta & 0 & sin\theta & 0 \\ 0 & 1 & -0 & 0 \\ -sin\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$TRotZ(\theta) = \begin{pmatrix} cos\theta & -sin\theta & 0 & 0 \\ sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Per invertir el sentit d'una transformació cal simplement invertir la matriu que relaciona els sistemes de referència.

Càlcul de transformacions

Inicialment es defineix una referència a la base del robot al centre geomètric de la base a l'altura del pla que compren tots els eixos de rotació dels actuadors. A partir d'aquesta es

defineix una referència a la plataforma a la seva posició de repòs a una distància H:

$$TPlat_i = Transl(0,0,H)$$

A continuació, es defineix la transformació que porta la referència $TPlat_i$ fins a la *pose* final. Aquesta *pose* desitjada depèn de com es vulgui controlar la pilota. En aquest cas, s'ha decidit que la plataforma roti entorn un eix paral·lel al pla de la plataforma en repòs i a una distància a aquesta igual al radi de la pilota (*r*). D'aquesta manera, si la pilota es troba a l'origen de la plataforma (el punt a on es vol estabilitzar aquesta per defecte), qualsevol moviment del robot afectaria el mínim possible al moviment de la pilota.

La seqüència de transformacions és aleshores:

- La translació inicial:

  $TPlat_i$

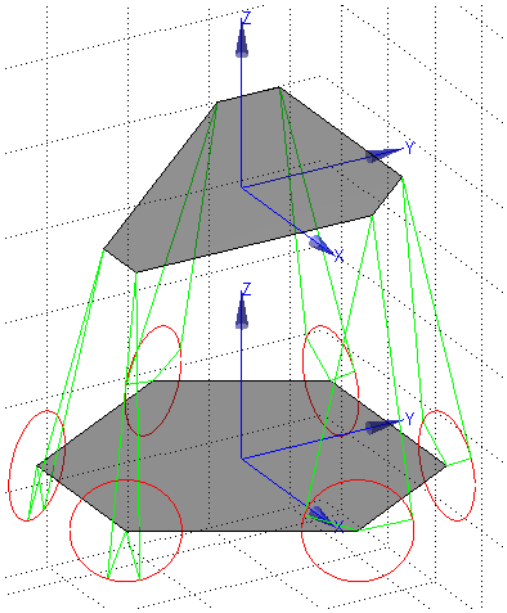


Fig 6.7. *Primera transformada representada en el simulador del robot.*

- Una translació a l'altura del radi de la pilota i el gir d'aquesta referència per tenir l'eix X situat en funció del comportament de la pilota en el pla de la plataforma (simplement és una rotació entorn a Z d'un cert angle α determinat pel control):

  $T_{eix} = Transl(0,0,r) \cdot TRotZ(\alpha)$

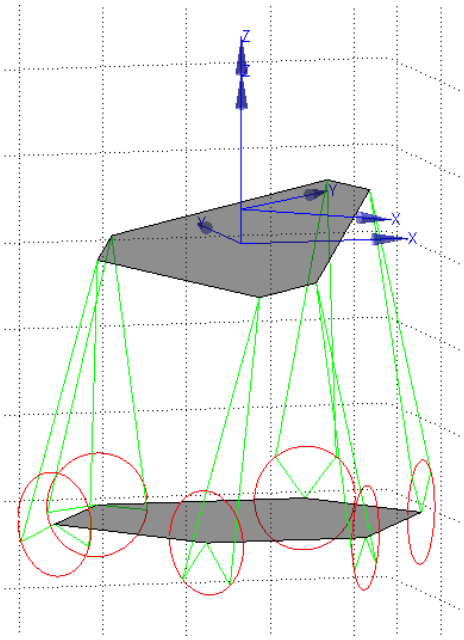


Fig 6.8. *Transformades del trasllat a distancia* r *i primera rotació del control representades en el simulador del robot.*

- Rotació entorn aquest eix X d'un valor θ determinat pel control per fer inclinar la plataforma:

  $TRotX(\theta)$

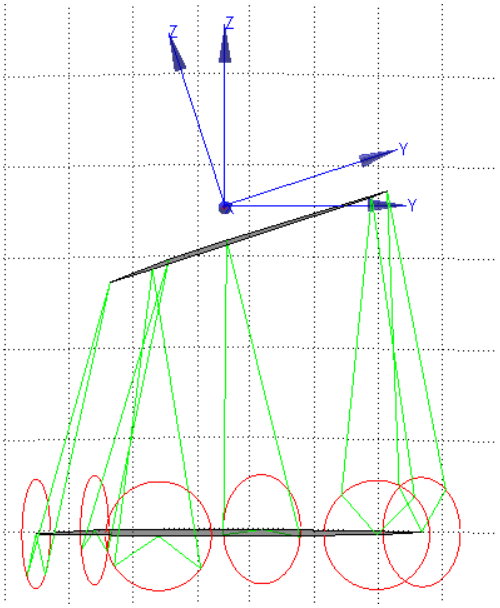


Fig 6.9. *Segona rotació del control representada al simulador*

- Una vegada rotada, cal tornar a referència plataforma mantenint la inclinació per poder calcular les coordenades dels punts d'unió de les potes a la plataforma. Aquestes coordenades es tenen en referència plataforma. Per tant:

T_{eix}^{-1}

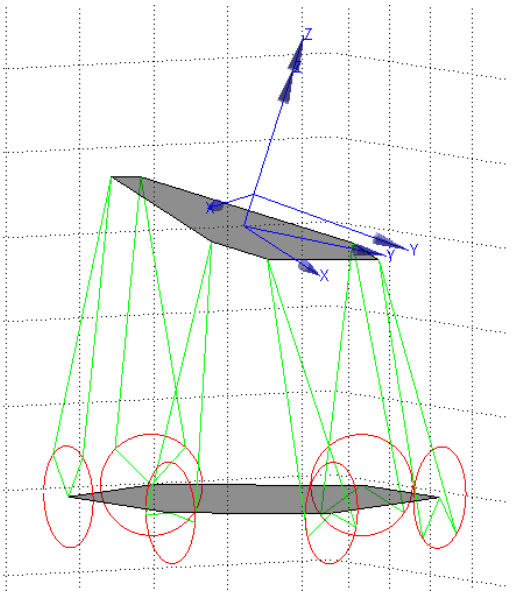


Fig 6.10. *Representació de la transformada per tornar a referencia plataforma.*

Finalment, doncs:

$$TPlat_f = TPlat_i \cdot T_{eix} \cdot TRotX(\theta) \cdot T_{eix}^{-1}$$

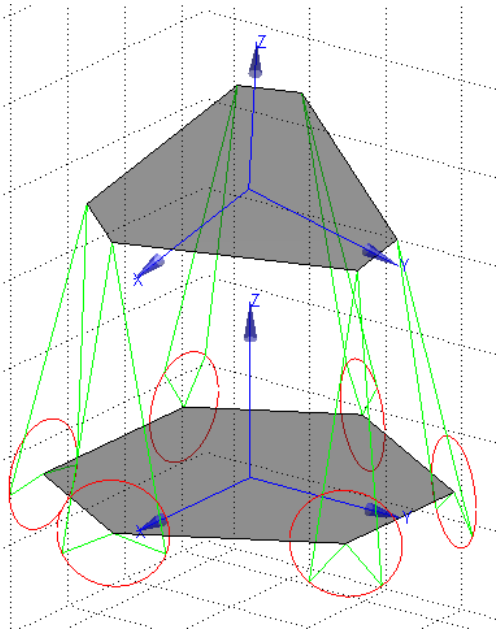


Fig 6.11. *Transformació final entre referencia base i referencia plataforma rotada representada al simulador*

Una vegada es tenen les dues bases ben definides l'una respecte l'altre, per resoldre caldrà situar els extrems de cada pota. Per tant, es defineixen 6 parelles de transformades de punts de la base i de la plataforma mòbil.

Es convenient que les referències de cada punt de la base (els origens de les quals pertanyen als eixos de rotació dels actuadors) tinguin un eix alineat amb el rotor ja que, com es veurà posteriorment, facilitarà els càlculs. Gràcies a les simetries de la base i la plataforma mòbil, només s'han d'introduir rotacions de 120º o 240º entorn a l'eix Z.

Així doncs, la definició de la transformada dels punts de la base es realitza de la manera següent:

$Tbase_1=Transl(L1, -RADI, 0)*TRotZ(-2\pi/3);$

$Tbase_2=Transl(-L1, -RADI, 0)*TRotZ(2\pi/3)$

A partir d'aquí, ens podem aprofitar de la simetria de l'estructura i afegir rotacions de terços de volta als dos punts prèviament calculats.

$Tbase_3=TRotZ(-2\pi/3)*Tbase1;$

$Tbase_4=TRotZ(-2\pi/3)*Tbase_2...$

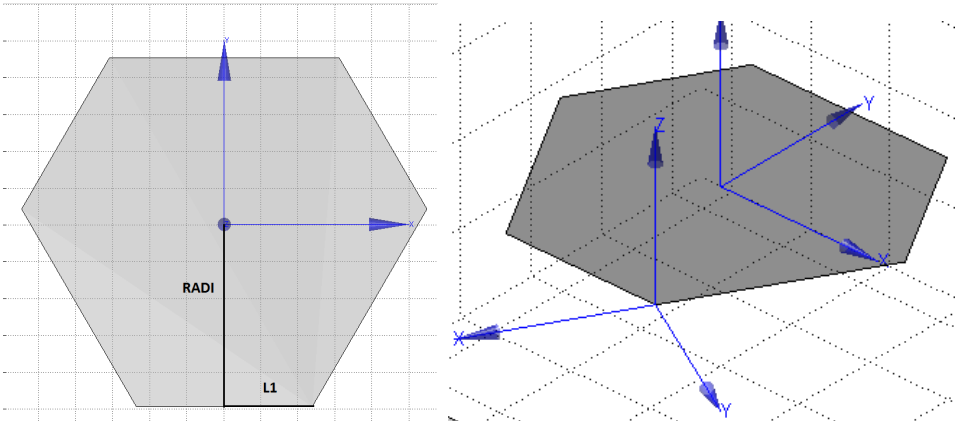


Fig 6.12. *Esquema de les distàncies definides en la base i la transformada final*

Pel que fa als punts de la plataforma, aquests estan relacionats amb el centre amb una transformada de translació pura, ja que no defineixen un pla com sí que fan els actuadors i l'orientació és torna irrellevant (recordem que la transformada al centre és $TPlat_f$):

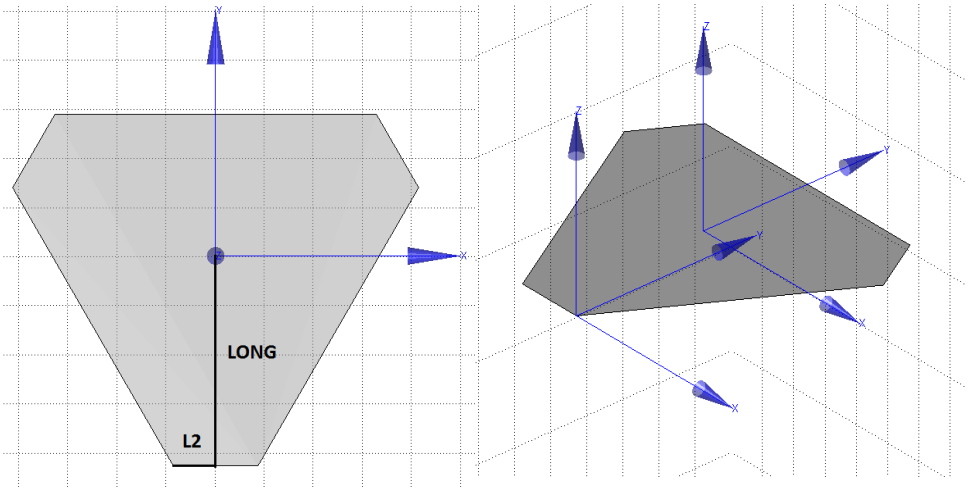


Fig 6.13. *Esquema de les distàncies definides en la plataforma la transformada final*

$Tplat_1$=Transl(L2, -LONG, 0});

$Tplat_2$= Transl(-L2, LONG, 0).

Els sistemes de referència associats a la resta de punts de la plataforma es poden obtenir, com abans, simplement rotant aquests.

Plantejamen i resolució de les equacions de clausura

Finalment es pot procedir al càlcul de les solucions. Per cada pota del robot, el procediment és el següent:

En primer lloc, s'expressa el punt de la plataforma en coordenades del punt de la base, per fer això cal recórrer a les transformades que els relacionen. El subíndex *i* indica el número de la pota del què es tracta:

$T_i = Tbase_i^{-1} \cdot Tplat_f \cdot Tplat_i$

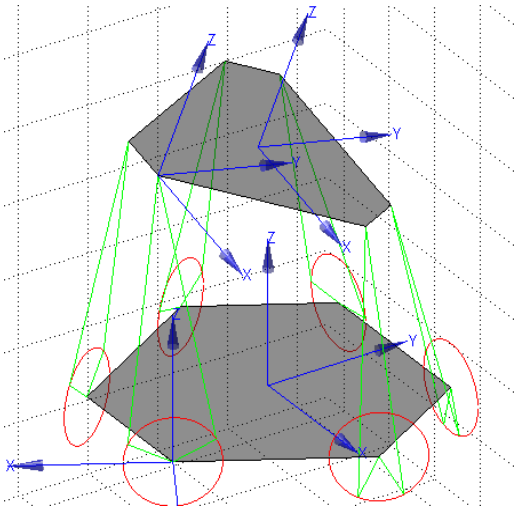


Fig 6.13. *Camí de transformades per expressar un punt de la plataforma en base punt base*

Després, es pot procedir a resoldre l'eqüació de clausura. En aquest cas les solucions possibles corresponen a la intersecció de la closca d'una esfera –deguda a la unió amb ròtules esfèriques als extrems– amb un cercle –degut als actuadors amb el braç.

Per tal de simplificar aquest càlcul, es treballa al pla definit pel braç de l'actuador corresponent. En aquest pla, les relacions geomètriques són les que es poden observar en la figura.

En la base de l'actuador, les coordenades del punt de la plataforma es troben amb:

$(x_i, y_i, z_i)^T = T_i \cdot (0, 0, 0, 1)^T$

I la correspondència amb *x* i *y* del diagrama és $x = x_i$ i $y = z_i$ degut a la orientació de la referència en l'actuador.

La variable *B* és directament la longitud dels braços dels actuadors i és constant.

L correspon a la projecció sobre el pla de la pota. Amb les tres coordenades del punt superior conegudes així com les relacions del punt lligat al braç es té que la projecció és:

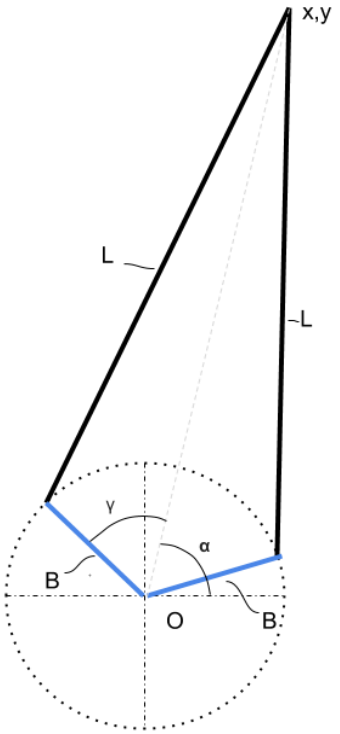


Fig 6.14. *Trigonometria de les equacions de clausura*

$$L = \sqrt{L_{pota}{}^2 - z_i{}^2}$$

On L_{pota} és conegut i z_i també.

Una vegada totes les constants són conegudes, cal determinar els dos angles de rotació possibles de l'actuador que resolen l'equació.

Aplicant el teorema del cosinus s'obtenen φ_1 i φ_2:

$$\varphi_1=\alpha+\gamma;\ \ \varphi_2=\alpha-\gamma;\ \ amb:$$

$$\alpha=arctan\left(\frac{y}{x}\right);$$

$$\gamma=arccos\left(\frac{B^2+x^2+y^2-L^2}{2\cdot B\cdot\sqrt{x^2+y^2}}\right)$$

Les equacions són anàlogues per cada pota del robot.

La tria de la solució d'entre les dues es basa en l'orientació inicial del braç de cada actuador. En la configuració inicial, el braç de cada actuador es troba en un dels dos costats de la circumferència i això marca quina solució cal prendre.

6.4. Programa de càlcul i simulador

Per tal de tenir una representació més entenedora del robot, de l'espai de treball i de la manera com enfocar el control, es realitzà un programa simulador amb Matlab®. Aquest programa, imposant les transformades descrites en l'apartat 6.3, és capaç de representar la resposta del robot davant d'una posició de la pilota i un angle de rotació donat. Emprant el càlcul simbòlic que ofereix Matlab®, a més a més, es podien obtenir les equacions que resolien la cinemàtica inversa i es podien ja aprofitar per calcular ràpidament la resposta dels actuadors per garantir l'orientació desitjada.

6.4.1. Robotics toolbox

Per resoldre la cinemàtica inversa del robot s'ha fet servir la *Robotics Toolbox* d'en Peter Corke. Aquesta toolbox consisteix en un conjunt de funcions i algorismes que condensen la teoria i la programació en entorn Matlab® de tot el relatiu amb la robòtica, la visió i el control. La llibreria, juntament amb el llibre que l'acompanya, cobreix en gran mesura els coneixements de la robòtica i la visió per computador modernes i permet enfrontar problemes complexos en poques línies de codi [24].

6.4.2. Pseudocodi

En aquest apartat es descriuen els passos que es fan servir a nivell informàtic per tal d'arribar a les solucions de la configuració del robot. El programa original és en Matlab® i es troba a l'annex. Aquest script a més a més, si s'executa amb el *Matlab Simbolic Toolbox*, permet trobar les 6 solucions en funció de les variables d'entrada i aquest codi es pot aprofitar posteriorment pel llaç de control.

RUTINA PRINCIPAL

```
%Definició de constants (en dm)
LONG = 0.8556;
L2 = 0.1723;
RADI = 0.9899;
L1 = 0.4750;
LEG = 1.68;
BRAC = 0.30;
H = 1.6;
R = 0.2;

ballangle = pi/4; % Aquest és l'angle del vector de posició de la pilota
en polars sobre el pla inicial
rotangle = pi/10; % Aquest valor ha de dependre de la distància de la
pilota al centre, es la inclinació de la plataforma entorn a l'eix de
rotació.

%TRANSFORMACIONS DE LA PLATAFORMA
RE = transl(0,0,H);
TAXIS = transl(0,0,R)*trotz(ballangle + (pi/2)); %Es suma pi mitjos a
l'angle de la pilota ja que la rotació per influir sobre aquesta s'entén
que ha de ser entorn a un eix perpendicular al vector posició
TR =RE*TAXIS*trotx(rotangle)*inv(TAXIS);

%DEFINICIO TRANSFORMADES DE PUNTS DE PLATAFORMA "TOP"
TP(:,:,1) = TR*transl(L2, -LONG, 0);
TP(:,:,2) = TR*transl(-L2, -LONG, 0);
TP(:,:,3) = TR*trotz(-2*pi/3)*transl(L2, -LONG, 0)
TP(:,:,4) = TR*trotz(-2*pi/3)*transl(-L2, -LONG, 0)
TP(:,:,5) = TR*trotz(-4*pi/3)*transl(L2, -LONG, 0)
TP(:,:,6) = TR*trotz(-4*pi/3)*transl(-L2, -LONG, 0)

%DEFINICIO TRANSFORMADES DE PUNTS DE LA BASE "BASE"
```

```
TB(:,:,1) = transl(L1, -RADI, 0)*trotz(-2*pi/3);
TB(:,:,2) = transl(-L1, -RADI, 0*trotz(2*pi/3);
TB(:,:,3) = trotz(-2*pi/3)*transl(L1, -RADI, 0)*trotz(-2*pi/3);
TB(:,:,4) = trotz(-2*pi/3)*transl(-L1, -RADI, 0)*trotz(2*pi/3);
TB(:,:,5) = trotz(-4*pi/3)*transl(L1, -RADI, 0)*trotz(-2*pi/3);
TB(:,:,6) = trotz(-4*pi/3)*transl(-L1, -RADI, 0)*trotz(2*pi/3);

%Coordenades de TOP point en referència BASE
Per i=1 fins i=6 fes
    %Definim una nova transformada
    TBP(:,:,i) = inv(TB(:,:,i))*TP(:,:,i);
    %Calculem els valors dels orígens en aquesta nova base
    Centers(:,i) = TBP(:,:,i)*[0; 0; 0; 1];
Fi per

%Càlcul de les solucions
Per i=1 fins i=6 fes
    [phi1, phi2]= SolveAngles(Centers(1,i),Centers(3,i),BRAC,sqrt(LEG^2-
    Centers(2,i)^2)); %SolveAngles descrit a continuació
    Sol1(i)=phi1;
    Sol2(i)=phi2;
Fi per

%Ara es tenen 2 vectors "Sol1" i "Sol2" amb les dues solucions, una a
cada vector

%Selecció de solucions i traducció a passos de l'actuador

%Degut a la distribució de les potes del robot alternades en direcció es
pren una o l'altre de manera alternada coincidint els punts descrits amb
el corresponent en la geometria real del robot. Aquesta decisió es pren
durant el muntatge. Es vol que els eixos definits en la plataforma siguin
paral·lels als de la càmera per facilitar els càlculs. Una vegada es
situa el robot respecte la càmera amb aquesta intenció cal comparar amb
el programat per fer coincidir les solucions amb els motors reals.
newSol = [Sol1(1),Sol2(2),Sol1(3),Sol2(4),Sol1(5),Sol2(6)];
```

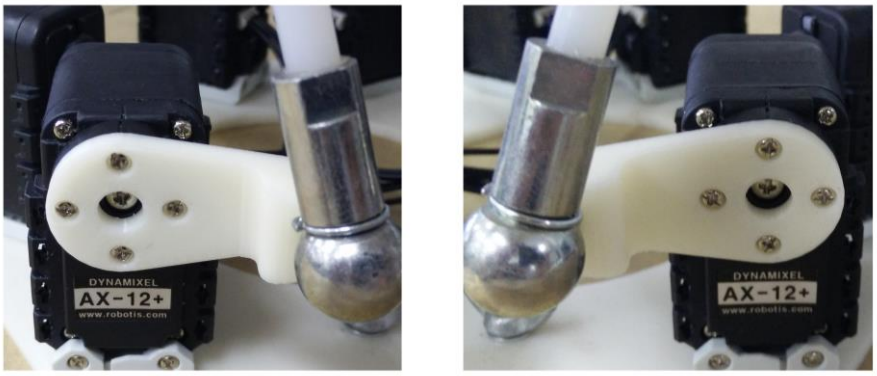


Fig 6.15. *Detalls de les posicions inicials dels actuadors*

Els braços del robot es situen alternats en direcció, però es situen en el punt mig del rang d'actuació de cada servomotor com s'aprecia a la figura 6.15 en 2 actuadors consecutius. Les marques en la carcassa del motor i en el rotor coincideixen en les dues. Això implica que cada motor requereix de la seva pròpia transformació d'angle solució a passos i direcció de rotació. Els angles estan definits des de l'horitzontal en coordenades de la referència establerta a cada actuador. Amb l'ajuda de les especificacions del motor i sabent que treballa en "Joint mode" amb un pas de 0,29º i 512 com a "offset" pel punt mig (0-1023) es resol:

```
parell = fals;
Per i = 1 fins i = 6 fes
    Si parell == fals fes
        %Es reescriu newSol
        newSol(i) = arrodoneix((-(newSol(i)-pi)*180/pi)/0.29)+512;
        par = cert;
    sinó fes
        %Es reescriu newSol
        newSol(i) = arrodoneix(-(newSol(i)*180/pi)/0.29)+512;
        parell = fals;
    Fi si
Fi per

%Finalment es prepara el vector de posicions dels motors ordenats per ID
send_Pose = [newSol(1),newSol(6), newSol(5), newSol(4), newSol(3),
newSol(2)];
```

SOLVE ANGLES

```
%Aquesta subrutina resol les equacions de clausura de forma analítica de
l'apartat 6.3
Funció [phi1, phi2] = SolveAngles(cx,cy,r1,r2)

%Es tria la funció arc tangent 2 de matlab que troba la solució de 4
quadratns per a valors X i Y reals atan2(X,Y)
alpha = atan2(cy, cx);
gamma = acos((r1^2 + cx^2 + cy^2 - r2^2)/(2*r1*sqrt(cx^2+cy^2)));

phi1 = alpha + gamma;
phi2 = alpha - gamma;

Fi funció
```

Usant la llibreria de càlcul simbòlic, fent variables les entrades del control: *ballangle* i *rotangle*, s'obtenen les expressions de les instruccions de cada motor. Degut a la complexitat de les solucions, que són expressions extremadament llargues, es reserven per l'annex.

6.4.3. Simulador

Amb l'ajuda de les solucions calculades i de tenir les dimensions dels elements del robot es va programar un simulador. Aquest simulador havia de ser una ajuda per representar i entendre el comportament del robot. També permetia observar les configuracions no accessibles ja que la representació resultant d'una configuració impossible queda delatada com es veu a la figura 6.16.

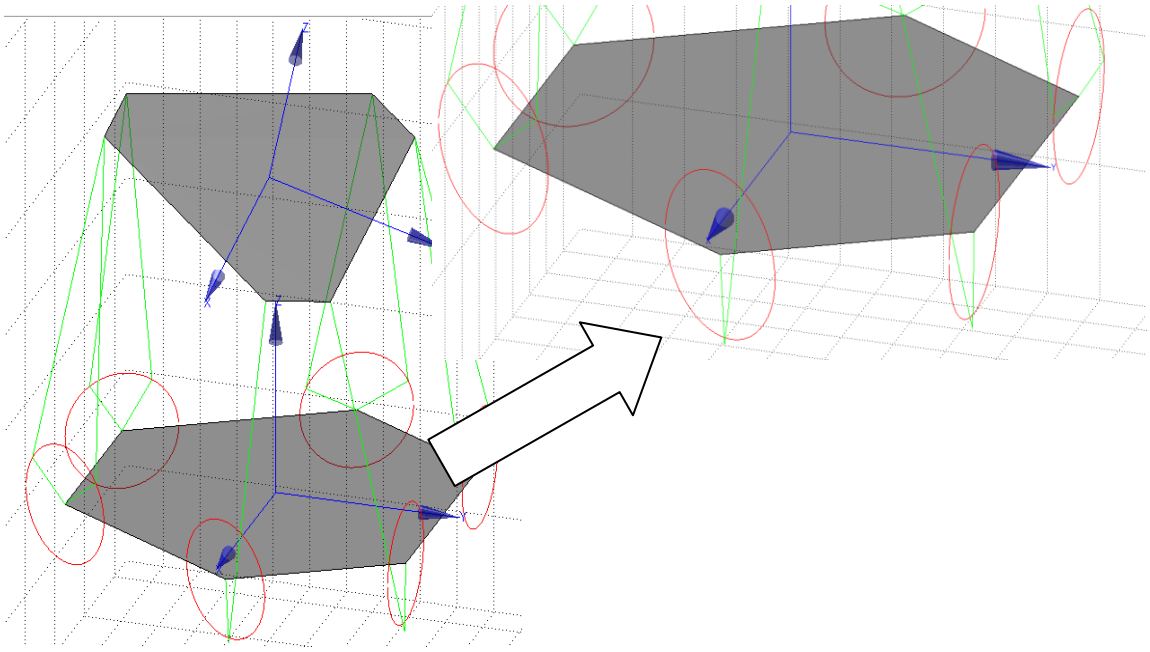


Fig 6.16. *Representació del resultat de simular una configuración inaccesible pel robot.*

En el cas representat, per tal de complir les restriccions cinemàtiques caldrien un braç i una pota de major longitud, el que acaba suposant resultats imaginaris en el càlcul de la instrucció per l'actuador ja que les longituds en qüestió són fixes. El simulador ha permès observar que no es adient superar els 18º (π/10) en qualsevol rotació entorn a l'eix a l'altura del radi de la bola.

El simulador també ha servir per il·lustrar la cinemàtica del robot en els apartats anteriors.

7. El control

El procés de mantenir la bola equilibrada en un punt del pla es pot considerar com la generalització a dues dimensions del problema de mantenir una pilota sobre un carril. El control de sistemes inestables sol presentar sempre un repte degut a no linealitats, i degut també al que el control ha d'ésser actiu constantment per garantir el comportament estable desitjat. De la mateixa manera, els sistemes multi-entrada multi-sortida amb acoblament presenten també problemes pel disseny del control.

En aquest problema de control en concret, els factors d'alta complexitat i el temps limitat van prioritzar el pragmatisme. En la literatura estudiada es fan servir les equacions de la dinàmica lagrangiana de la pilota en funció dels parells provinents de la plataforma i aquesta és relacionada amb els parells dels actuadors. Aquest enfocament es troba sobretot en plataformes de dos graus de llibertat, on la dinàmica és més senzilla i existeix una variable de control dels actuadors amb una relació coneguda amb el parell produït. Amb les equacions de l'esmentat prèviament i models matemàtics pels sensors (en aquest cas, visió), es pot procedir a un control molt més acurat ajustat matemàticament.

Els 6 graus de llibertat del robot paral·lel, el controlador intern no ajustable dels servos AX-12+ i la variació constant de superfície, de pilota, etc; van requerir d'un ajustament empíric. Amb el mètode de Ziegler-Nichols per anell tancat i controladors PID, propis del control lineal, es va descompondre el moviment de la plataforma menyspreant els acoblaments del sistema. En aquest apartat es tracta l'estratègia de control, l'arquitectura del llaç i finalment del programa de control de Simulink®.

7.1. Estratègia

Seguint el ja introduït al punt 6.3, per influir en la pilota i fer-la desplaçar en la direcció desitjada, es fa rotar la plataforma un cert valor entorn un eix que pertany i talla el centre del pla definit anteriorment. La determinació d'aquests és la tasca del control i calia definir els criteris a partir dels quals el sistema havia de prendre decisions. En la tria d'aquests criteris es varen seguir diferents estratègies.

7.1.1. Actuació en coordenades polars

En primer lloc, es va plantejar el problema en coordenades polars. Es va creure que aquest enfocament simplificava la manera de controlar la pilota. Mitjançant la posició d'aquesta en la plataforma en coordenades cartesianes provinents de la visió i transformant-les en polars, es

podia determinar l'eix de rotació com a perpendicular al vector de posició en el pla de la plataforma en repòs. Així, es controla únicament la rotació amb un PID que actua sobre la variable distància *r*.

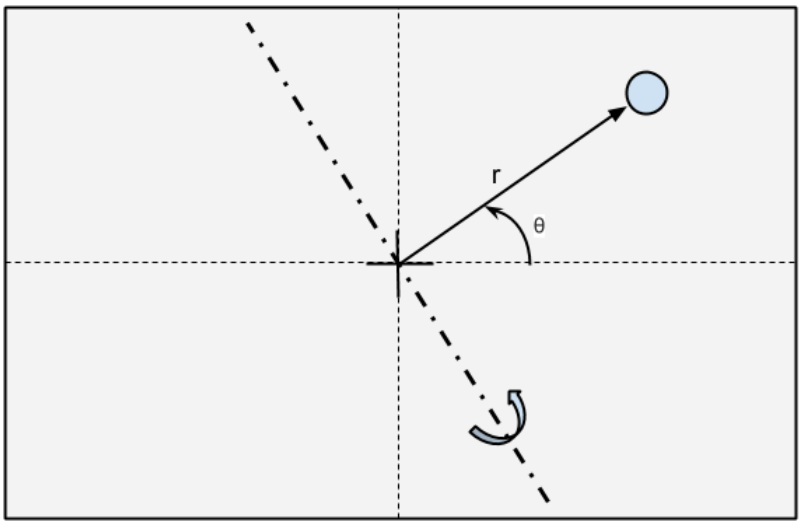


Fig 7.1. *Esquema general de l'estratègia de control.*

En aquesta configuració el PID controlava únicament un grau de llibertat: distància al centre. L'eix quedava definit directament com a perpendicular al vector posició rotant $\theta+\pi/2$ i pertanyent al pla descrit en la cinemàtica. Fou degut a aquesta estratègia de control que es va definir el càlcul de rotacions com s'exposa als apartats 6.3 i 6.4 i que van fer de *rotangle* i *ballangle* les variables de control.

Aquesta estratègia va revelar un conjunt de mancances:

- Evidentment, no es podia controlar cap trajectòria programada de la pilota al pla.
- Per a trajectòries inicials de la pilota que no creuessin el centre de la plataforma, si la velocitat era suficient, la bola orbitava i accelerava en espiral creixent fins sortir de la plataforma. Aquest fenomen és degut a que no tenim cap control sobre el comportament d'θ.
- El temps a l'equilibri era molt variable, en part degut al punt anterior.

Per tot això, i sense haver-hi una opció còmoda per controlar velocitats i posicions angulars, es va decidir investigar en una altra direcció, descomponent els moviments entorn dos eixos i duplicant el nombre de controladors.

7.1.2. Rotació entorn a dos eixos

En la rotació entorn a dos eixos es pretenia resoldre les mancances del sistema de control anterior, tot aprofitant el programa de resolució de la cinemàtica inversa amb variables *rotangle* i *ballangle*.

En els casos estudiats en la literatura, ja que sovint es té el control sobre dos eixos perpendiculars de la plataforma i les equacions dinàmiques són no lineals i acoblades, es procedeix a una linealització entorn al punt d'operació per tal de poder aplicar tècniques de control lineal. A més a més, es pot assumir que per a petites rotacions de la plataforma des de la horitzontal, el sistema serà desacoblat. L'anàlisi d'aquesta manera de fer es troba molt ben definida a [1]. Cal remarcar que aquest acoblament sovint es troba en un dels dos angles ja que les rotacions no són operacions commutatives.

Seguint amb aquesta filosofia i essent conscients que la dinàmica de la plataforma Stewart és ben diferent a la d'una plataforma de 2 graus de llibertat on cada rotació va associada a un actuador, sí que era útil assumir la commutativitat de dues rotacions consecutives d'un pla per als angles del rang d'operació.

El plantejament, doncs, va consistir en definir unes rotacions entorn a dos eixos del pla de rotació paral·lel i compondre una rotació final. Una rotació entorn un únic eix d'aquest pla amb un únic valor de rotació.

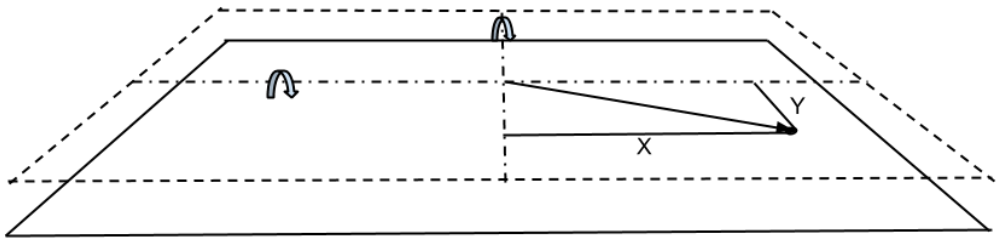


Fig 7.2. *Esquema general de la segona estratègia de control estudiada*

D'aquesta manera doncs, cada rotació era determinada en funció de l'error de la coordenada de posició perpendicular a l'eix de rotació (és l'eix que té la influència sobre el desplaçament de la bola en aquesta direcció). Amb un controlador actuant sobre cada eix és clar que es podrà realitzar un control molt millor ja que es dupliquen els paràmetres a determinar (3 constants a cada controlador).

Per determinar aquesta rotació final es realitza un càlcul basat en quaternions ja que l'extracció dels valors d'interès es tornava més còmoda.

Quaternió

Una rotació a l'espai es pot representar mitjançant un quaternió unitari de la forma:

$$Q = \cos(\theta/2) + w_x . \sin(\theta/2)\, i + w_y . \sin(\theta/2) j + w_z . \sin(\theta/2)\, k,$$

a on *i*, *j* i *k* son unitats imaginaries, (w_x , w_y , w_z) representa el vector unitari en la direcció de l'eix de gir, i θ és l'angle rotat.

En general, el producte de dues rotacions no és commutatiu. És en aquest punt on es realitza l'aproximació per a petites rotacions com s'ha descrit en la introducció d'aquest tema. Això vol dir que l'ordre en que es composin ambdues rotacions variarà el resultat final, amb un error que s'ha considerat assumible pels rangs de treball.

Es important adonar-se que la correspondència entre rotació tridimensional i quaternió no es única. Una rotació en 3D es pot representar amb dos quaternions diferents depenent del sentit escollit per definir l'eix de rotació.

El producte de dos quaternions, $a_1 + b_1 i + c_1 j + d_1 k$ i $a_2 + b_2 i + c_2 j + d_2 k$, anomenat producte Hamiltonià $(a_1 + b_1 i + c_1 j + d_1 k)\ (a_2 + b_2 i + c_2 j + d_2 k)$, es determina pel producte de les components de la base seguint una llei distributiva. El producte es torna una suma d'elements de la base, que agrupats resulta:

$$a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 + (a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2) i + (a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2) j + (a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2) k.$$

Resolució

Per la resolució de la rotació final, cal definir els dos eixos de rotació del pla i rebre les instruccions provinents del control per a les rotacions entorn de cadascun. Una vegada obtinguts es realitza el producte Hamiltonià entre ells en l'ordre que es vulgui. Es sap que el resultat variarà però aquesta variació es petita en la mesura en que les rotacions son tetites. Finalment s'extreu l'eix de rotació i l'angle girat del vector final. Qualsevol producte de rotacions d'eixos prependiculars d'un mateix pla, resultarà en una rotació amb una component en Z. Aquest fet és evident degut a que, en realitzar la primera rotació, el segon eix experimenta un moviment fora del pla inicial inevitable.

En la cinemàtica inversa plantejada, per simplificar, el moviment només es pot realitzar

entorn d'un eix pertanyent al pla inicial. És per això que finalment es projectarà l'eix resultant sobre d'aquest, prenent les dues primeres components del vector.

Dins del programa de Simulink® es convertiran les expressions obtingudes en les variables de control de la rotació *ballangle* i *rotangle.*

Altre cop, la rutina de Matlab® per trobar l'expressió final de l'angle rotat i l'eix de rotació, es presenta una idea ordenada del mètode de resolució. Les variables determinades pel control són X_{rot} i Y_{rot} i recordem que es determinen en funció de l'error mesurat.

```
%Definició dels eixos de rotació, és directe
Qx = [cos(xrot/2), sin(xrot/2), 0 , 0];
Qy = [cos(yrot/2), 0, sin(yrot/2), 0];

%Càlcul del producte Hamiltonià
Qrot = [Qx(1)*Qy(1)-Qx(2)*Qy(2)-Qx(3)*Qy(3)-Qx(4)*Qy(4),...
        Qx(1)*Qy(2)+Qx(2)*Qy(1)+Qx(3)*Qy(4)-Qx(4)*Qy(3),...
        Qx(1)*Qy(3)-Qx(2)*Qy(4)+Qx(3)*Qy(1)+Qx(4)*Qy(2),...
        Qx(1)*Qy(4)+Qx(2)*Qy(3)-Qx(3)*Qy(2)+Qx(4)*Qy(1)];

%Es determina l'angle de rotació final com components i tangents de
l'angle rotat.
rotaxis = (1/Qrot(1))*[Qrot(2),Qrot(3),Qrot(4)];
%Es normalitza
norm_rotaxis = norm(rotaxis);
rotaxis = rotaxis/norm_rotaxis;
%Amb el mòdul es pot extreure directament el resultat amb la funció arc
tangent
rotangle = atan(norm_rotaxis)*2;
```

I s'obté que la rotació final ve donada per:

```
rotangle = 2*atan((abs(sin(xrot/2))^2/abs(cos(xrot/2))^2
           +abs(sin(yrot/2))^2/abs(cos(yrot/2))^2
           +abs(sin(xrot/2)*sin(yrot/2))^2/(abs(cos(xrot/2))^2
           *abs(cos(yrot/2))^2))^(1/2))
```

I les tres components del vector unitari en la direcció de l'eix de gir per:

```
Component X = sin(xrot/2)/(cos(xrot/2)
           *(abs(sin(xrot/2))^2/abs(cos(xrot/2))^2
           + abs(sin(yrot/2))^2/abs(cos(yrot/2))^2
           + abs(sin(xrot/2)*sin(yrot/2))^2/
           (abs(cos(xrot/2))^2*abs(cos(yrot/2))^2))^(1/2)),

Component Y = sin(yrot/2)/(cos(yrot/2)
           *(abs(sin(xrot/2))^2/abs(cos(xrot/2))^2
           + abs(sin(yrot/2))^2/abs(cos(yrot/2))^2
           + abs(sin(xrot/2)*sin(yrot/2))^2/
           (abs(cos(xrot/2))^2*abs(cos(yrot/2))^2))^(1/2)),
```

```
Component Z = (sin(xrot/2)*sin(yrot/2))/
              (cos(xrot/2)*cos(yrot/2)
              *(abs(sin(xrot/2))^2/abs(cos(xrot/2))^2
              + abs(sin(yrot/2))^2/abs(cos(yrot/2))^2
              + abs(sin(xrot/2)*sin(yrot/2))^2/
              (abs(cos(xrot/2))^2*abs(cos(yrot/2))^2))^(1/2))]
```

7.2. Arquitectura del llaç

Per realitzar el control es va voler recórrer a les tècniques de control lineal, que eren les que tenien més sentit en un sistema de control no basat en el model matemàtic del robot i la pilota. Dada la potència dels controladors PID, no es va dubtar en emprar aquesta classe de control seguint també la feina feta per [17]. Un altre argument per fer servir aquests controladors fou que eren els que presentaven un mètode d'ajustament empíric molt conegut i estudiat com el de les oscil·lacions mantingudes o Ziegler-Nichols.

Aquesta tria va fixar l'arquitectura en situar els controladors PID en el senyal d'error, ja que és on està definida la metodologia per ajustar-los.

El freqüència de mostratge del sistema ve imposada per la màxima velocitat de la càmera en aquesta resolució i és de 60Hz (fotogrames per segon).

7.2.1. Controladors PID

Els controladors PID s'anomenen així degut a les seves 3 accions: proporcional, integral i derivativa. És un tipus de controlador de sistema realimentat àmpliament utilitzat en el camp industrial. Un sistema PID calcula un valor error entre una mesura de la sortida del sistema i un punt d'operació desitjat. L'objectiu del controlador és manipular una variable interna del sistema per minimitzar o anul·lar l'error. També pot influir en la resposta dinàmica en la transició entre estats d'equilibri.

Explicat de manera intuïtiva, cada component del controlador regula la variable tenint en compte tres aspectes temporals de l'error:

- P: Amb la constant proporcional s'actua en funció de l'error actual.
- I: La part integradora actua en funció de l'històric de l'error.
- D: El control derivatiu actua en previsió de l'error futur basant-se en el ritme de canvi de l'error actual

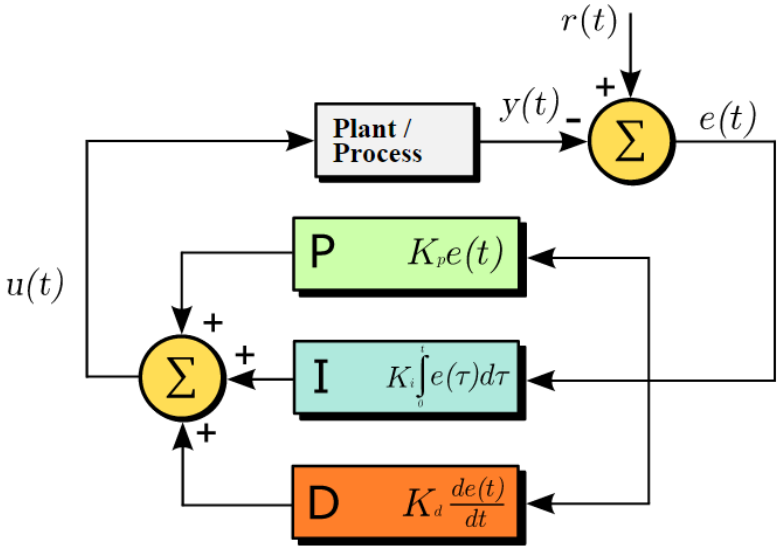


Fig 7.3. *Diagrama de funcionament de la realimentació d'un PID amb les expressions temporals [25]*

La suma ponderada d'aquestes tres components és la feta servir per ajustar el sistema mitjançant la variable de control. El fet que el control només actuï sobre una variable sense conèixer la resta del procés el fa extremadament versàtil.

Algunes aplicacions poden requerir l'ús de només alguna de les accions o de qualsevol combinació de parelles d'aquestes. Això s'aconsegueix fent 0 la constant corresponent a les parts del controlador que es vulguin anular. En el cas que ens ocupa, durant l'experimentació de seguida es va fer palesa la necessitat de les 3 components. Presentant les següents problemàtiques:

- P: Temps d'establiment molt alt i un rang de velocitats de pilota controlables molt baix.
- PD: Bon temps d'establiment i actuació però l'existència d'un error permanent deslluïa la resposta.
- PI: Mancat de velocitat de resposta i rang de velocitats controlables molt petit.

Tot i que el mètode de les oscil·lacions mantingudes permet aquestes arquitectures, l'ajustament final es va realitzar sobre les 3 accions del controlador.

7.2.2. Sintonía dels PIDs

Un bon ajustament dels paràmetres és imprescindible per tal que el sistema de control presenti la resposta desitjada. Existeixen les següents metodologies d'ajustament:

Sintonia analítica

Consisteix en assimilar el sistema obtingut mitjançant modelització matemàtica en sistemes de 1er o 2on ordre lineals. Realitzar una assignació de pols a aquests o calcular els coeficients que minimitzin els índex desitjats de la resposta.

Assimilar a sistemes de 1er o 2on ordre implica traduir les especificacions en característiques pròpies de sistemes d'aquest tipus: e_{ss}, T_p, S_p... A partir d'aquests es calcula una planta de l'ordre adequat amb les característiques desitjades i es realitza una assignació de pols. Aquest procediment també es pot realitzar amb taules de minimització d'índex.

El mètode parteix de tenir a disposició models matemàtics de la resposta i dels elements del control.

Sintonia manual

La sintonia manual consisteix en posar el sistema en llaç tancat i començar a canviar els valors del controlador de manera manual.

En primer lloc es realitza únicament una acció proporcional que s'augmenta fins a tenir una resposta acceptablement ràpida tot que l'error estacionari sigui no nul. A continuació s'augmenta l'acció integral de manera progressiva fins eliminar l'error estacionari en un temps adequat. Finalment s'incorpora l'acció derivativa fins recuperar la forma desitjada en la resposta transitòria.

Aquest mètode és el menys rigorós i sovint recau en l'experiència de l'encarregat de sintonitzar el sistema.

Sintonia empírica

La sintonització es realitza tant en anell obert com per sistemes en anell tancat.

La metodologia en anell obert o anomenada també Ziegler-Nichols en anell obert és un mètode que parteix de l'aproximació de la planta a un sistema de primer ordre amb K, T i T_0.

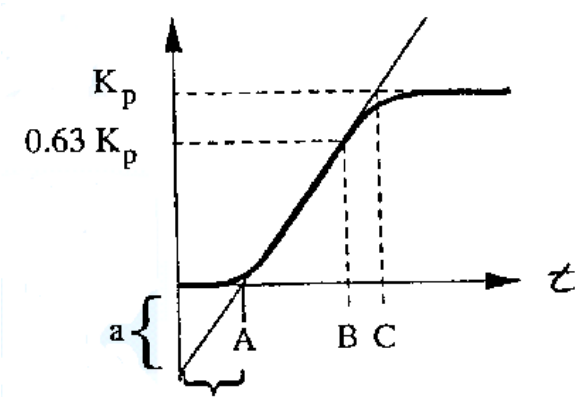


Fig 7.4. *Resposta inicial*

Consisteix en aplicar una entrada graó a la planta, deduir-ne les característiques de sistema de primer ordre equivalent i ajustar-lo segons la taula 7.1:

Controlador	K	T_i	T_d
P	1/a	-	-
PI	0,9/a	3L	-
PID	1,2/a	2L	L/2

Taula 1 Càlcul de paràmetres en funció d'a, L i arquitectura

Dat que el sistema amb el que es tracta en aquest projecte és inestable era impossible obtenir una resposta graó amb un valor final definit. La pilota sortia de la plataforma degut a la inclinació mantinguda.

La metodologia en llaç tancat, també anomenada de Ziegler-Nichols, consisteix en posar el sistema en anell tancat i control proporcional. D'aquesta manera, augmentar el valor de K de forma suau i progressiva fins la resposta oscil·lant mantinguda. Això implica en un diagrama polar trobar el segon tall amb l'eix de les abscisses.

En aquest punt, es determina el valor de K_u (K última abans d'inestabilitzar el sistema) i el període d'oscil·lació de la resposta del sistema en aquest punt T_u

.

Finalment ajustar el controlador d'acord amb la taula 7.2.

Controlador	K	T_i	T_d
P	$0,5K_u$	-	-
PI	$0,4K_u$	$0,8T_u$	-
PID	$0,6K_u$	$0,5T_u$	$0,12T_u$
PID 0%S_p	$0,2K_u$	$0,5T_u$	$0,33T_u$

Taula 2 Càlcul dels paràmetres del control amb el mètode d'anell tancat

Cal afegir que el mètode triat de les oscil·lacions mantingudes assumeix un model del sistema com:

$$\frac{K \cdot e^{-sT}}{(a+s)}$$

Fet que el fa una eina amb limitacions pel que fa a aplicacions i resultats obtinguts. Tot i això, es considerat un sistema molt adient per rebutjar pertorbacions aplicades al sistema.

Aquesta metodologia va resultar la més adient pel cas d'estudi. Permetia ajustar els paràmetres per un control on no es podia analitzar adequadament la resposta a un graó i permetia un rigor superior en l'obtenció de paràmetres d'un controlador desvinculat del model matemàtic. Tot i que només s'estudia un rang de la corba polar del sistema, de seguida va proporcionar resultats satisfactoris.

7.2.3. Sintonía i ajustament

Suposant que la dinàmica de la plataforma en els dos eixos entorn els quals es pretenia rotar és gairebé igual, es realitza un ajustament pel mètode de les ocil·lacions mantingudes. Aquest ajustament es realitza pel muntatge final amb la càmera, binarització, i esquema de control definitiu. Cal que sigui així, ja que qualsevol modificació de temps de mostreig o dels blocs del sistema farien que un mateix controlador presentés resultats diferents.

Cada controlador farà la seva acció sobre una component de l'error, així que es realitza un muntatge de carrils sobre la superfície per poder ajustar la pilota en un únic eix i aplicar els resultats a ambdós controladors (veure figura 7.5).

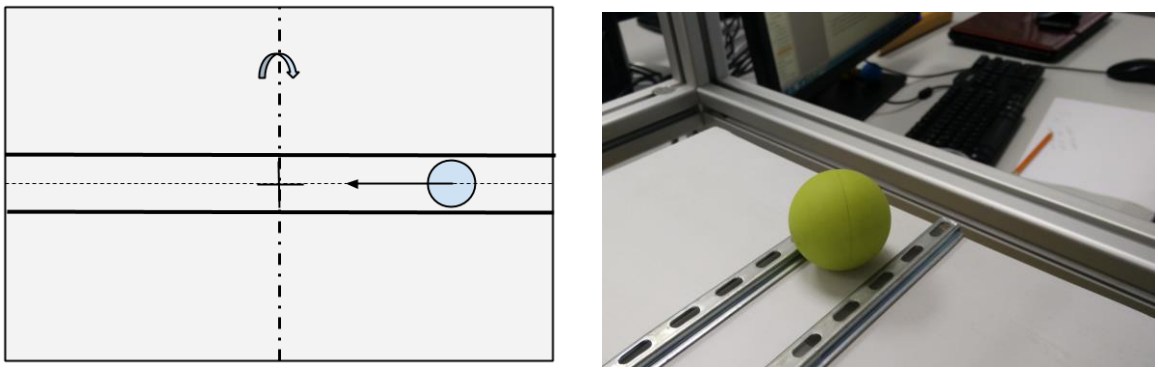


Fig 7.5. *Esquema del muntatge per la cal·libració. Fotografia amb el detall.*

En el muntatge i l'experimentació es va poder observar com les K_u per oscil·lació mantinguda variaven en funció de l'amplitud que se'ls donava inicialment. És a dir, una K_u per a 5 centímetres d'amplitud sobre la plataforma no presentava oscil·lacions mantingudes per a amplituds menors, sinó que tendia a l'equilibri. De fet, era molt més fàcil estabilitzar el sistema amb un control proporcional prop del centre i els canvis d'aquesta havien de ser molt notables per canviar la resposta. En canvi, aquesta mateixa K_u, per a amplituds majors accelerava la pilota i la treia de la plataforma inestabilitzant el sistema. A més a més, aquest es mostrava molt més sensible als canvis de K i al soroll, dificultant molt l'ajustament. Aquests fenomens es van atribuir al fet d'estar linealitzant entorn a un punt d'operació i tenir una dinàmica canviant conforme la pilota s'allunyava.

Es va decidir aplicar el mètode per una distància mitjana de 7,5 cm del centre per obtenir els millors resultats. Les mesures de temps per T_u es van obtenir amb la mitjana de la mesura de 10 preíodes consecutius mesurats amb Simulink® i l'ajuda del sistema de visió.

Per aquesta distància es va obtenir:

K_u = 2,665, que garantia les oscil·lacions mantingudes.

T_u = 1,125s de període d'oscil·lació.

Aplicant la taula del mètode de Ziegler Nichols per anell tancat amb una modificació basada en la resposta amb 0% de sobrepuig [26]:

$K_P = 0{,}2 \cdot K_u = 0.533$; $K_I = K_p \cdot T_i = 0{,}533 \cdot 0{,}5 \cdot T_u = 0{,}300$; $K_D = K_P \cdot T_D = 0{,}533 \cdot 0{,}33 \cdot T_u = 0{,}1980$;

Finalment, es vol destacar que es va afegir un *reset* de la funció integradora del controlador per no acumular l'error que es creava durant els moviments manuals de la pilota, pels quals seguia dins del sistema ja que la càmera n'observava la posició i després influien en la resposta. També es van saturar les sortides dels controladors per no excedir la configuració màxima de rotació del robot (18º) tal com s'explica a l'apartat 6.4.3.

7.3. El llaç en Simulink®

La integració de tots els elements del sistema de control es va realitzar en un únic programa de Simulink®. En aquest apartat s'analitza l'estructura general d'aquest llaç i el flux de la informació, així com la funcionalitat de cada bloc. El codi corresponent a les funcions de Matlab® de dins del programa es poden trobar a l'annex. La millor manera de descriure el programa és recorrent el camí de les dades en cada iteració.En l'esquema a pàgina sencera al final d'aquest capítol es pot apreciar la visió completa del programa.

Opcions de simulació

El control es va resoldre amb un *time-step* igual que el del sensor de 1/60 s. Es va resoldre amb el mètode numèric Runge-Kutta ordre 4 sense cap diferència notable en el canvi de mètode de resolució numèrica. Tots els blocs del sistema són de tipus discret amb el mateix període de mostratge per mantenir la coherència del control.

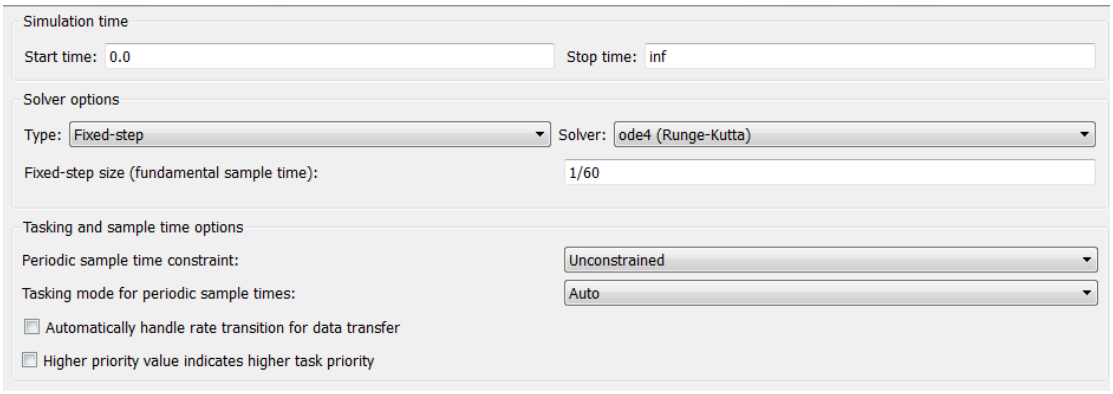


Fig 7.6. *Condicions de resolución del programari*

Visió

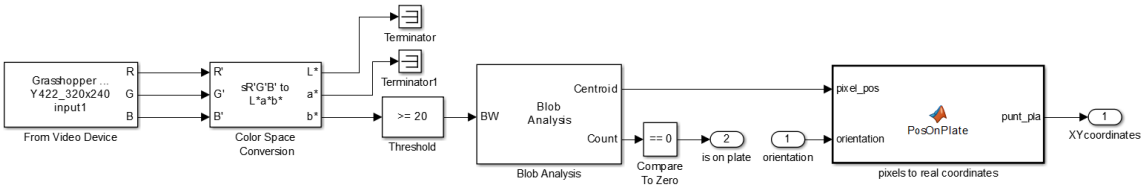


Fig 7.7. *Esquema de visió complet*

És el primer pas de processat de les dades del sensor (càmera) i es descriu amb detall a l'apartat 5.3.2 i 5.4.3. En tot aquest subsistema les dades entren com a imatges en color i en surten pels ports.

El port de sortida *XYcoordinates* conté un vector amb les coordenades X i Y del centroide de la pilota en el pla de la plataforma o bé en un pla imaginari paral·lel a la base i a la plataforma en repòs. Si rep pel port d'entrada *orientation* les variables de l'estat de l'instant anterior de la plataforma és quan projecta la posició en la plataforma.

El port de sortida *is on plate* correspon a un valor booleà que indica si la càmera té dins del camp de visió la pilota, pel qual val 0, o no i val 1.

Controladors

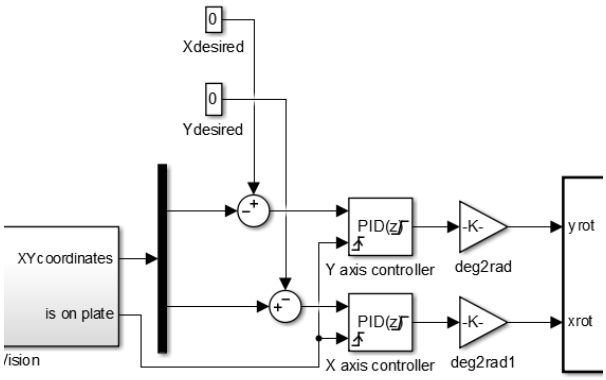


Fig 7.8. *Secció de control*

Les coordenades del punt en el pla es separen en dos camins i van als operadors de suma. Aquests calculen l'error absolut entre els valors desitjats (fixos en el temps o de variables degut a una trajectòria imposada) i la posició actual de la pilota.L'error segueix cap als controladors.

Els controladors són PID de temps discret amb els paràmetres descrits (vegeu la figura 7.9). L'error en la coordenada X entra al controlador de rotació de Y i viceversa. Aquests controladors ja treuen en graus la rotació, i cal convertir-la a radians per les operacions posteriors amb el bloc de guany (*gain*). La sortida està saturada en 18º de valor absolut de rotació pels dos eixos, ja que es vol prevenir qualsevol ordre per una configuració no accessible.

Pel que fa a la senyal binària de si la pilota és o no sobre la plataforma, el flanc de pujada d'aquesta fa el *reset* de l'error acumulat per l'acció integradora dels controladors. És equivalent a començar de nou cada cop que es treu del camp de visió la pilota.

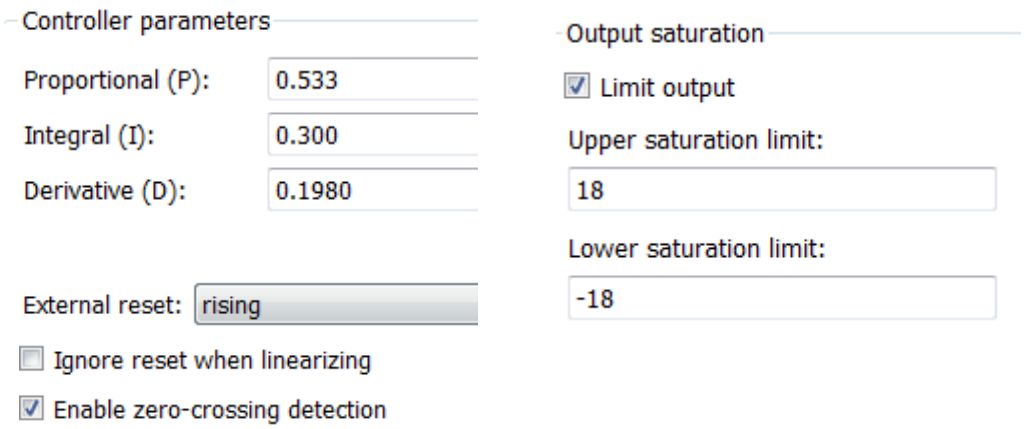


Fig 7.9. *Establiment dels paràmetres dels controladors per obtener el comportament desitjat.*

Càlcul d'orientació

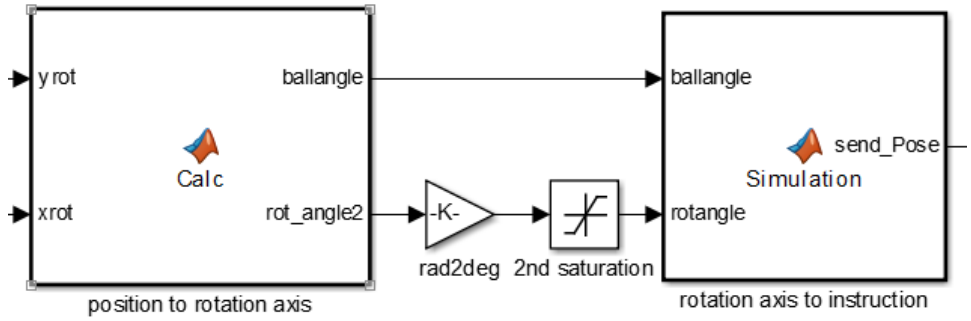


Fig 7.10. *Secció del càlcul de l'orientació.*

En aquesta part del programa es realitzen els càlculs corresponents al càlcul de la nova orientació del robot.

Position to rotation axis: En aquesta subrutina es calculen les instruccions de *ballangle* i *rotangle*. Fent servir les solucions analítiques descrites en l'apartat 7.1.2 calcula l'angle en polars d'orientació de la pilota amb les components d'X i Y i de la rotació calculada. La rotació entorn a l'eix és directament el resultat de l'expressió de *rotangle*.

A continuació es realitza una nova saturació de 18º per evitar que la composició de les dues rotacions pugui resultar en una configuració impossible i com a capa de protecció davant d'errors que es van trobar durant l'experimentació.

Rotation axis to instruction: Finalment es calculen les instruccions a enviar als actuadors. És simplement el càlcul analític de les 6 components del vector a partir de les variables d'entrada. És el resultat del càlcul descrit a l'apartat 6.3 i 6.4.

Escriptura als motors

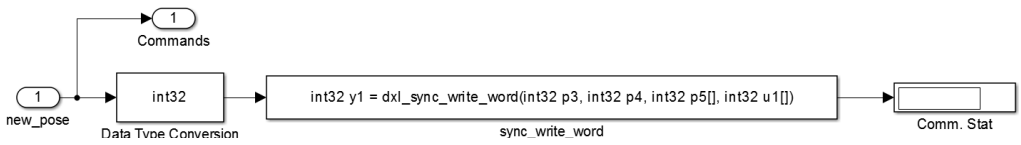
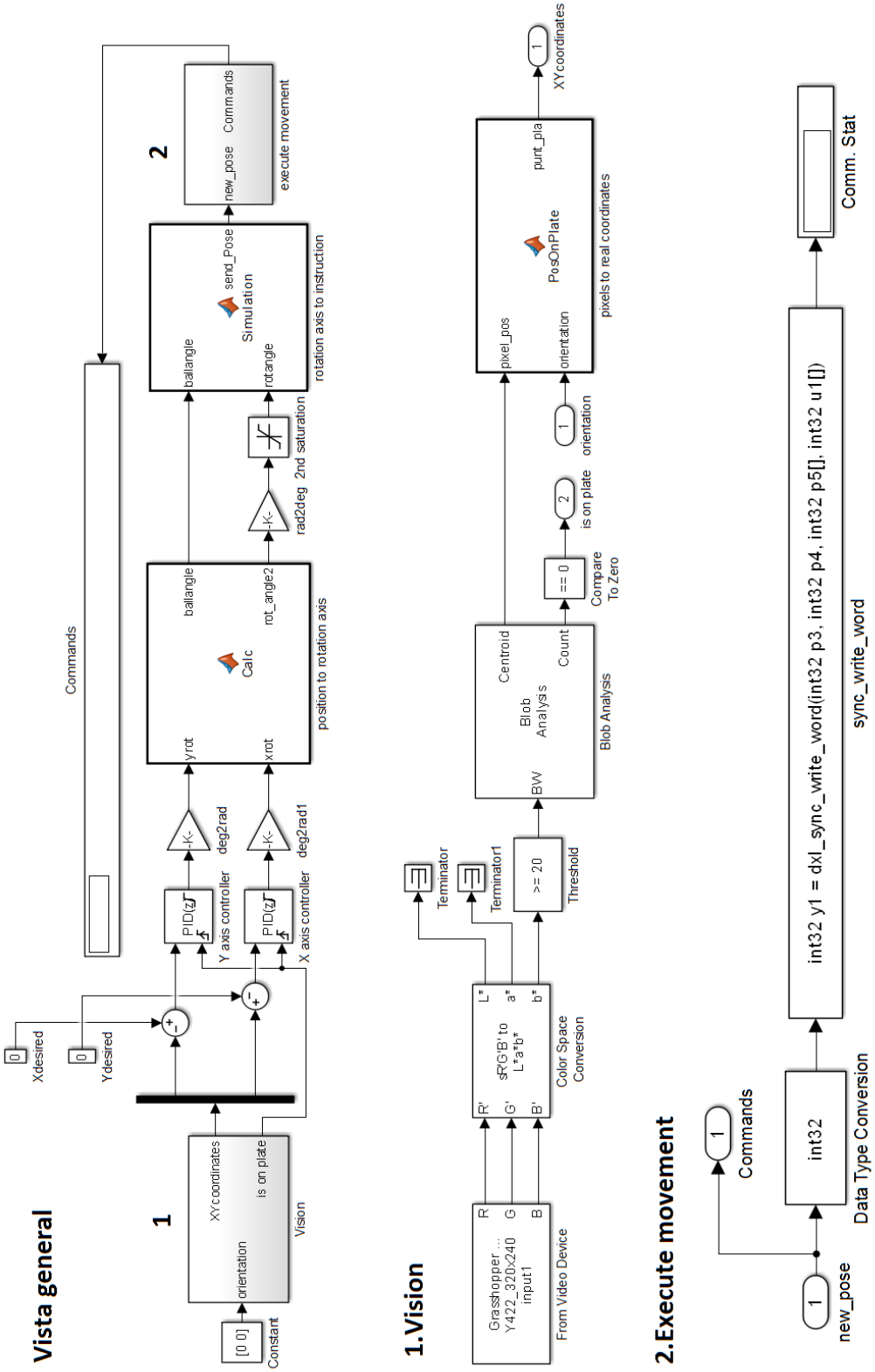


Fig 7.11. *Vista completa de la secció del programa dedicada a escriure als actuadors*

Finalment s'escriu a les adreces de posició dels motors mitjançant el bloc de comunicació que es va programar i implementar en Simulink® i està explicat en l'apartat 4.4 i concretament es tracta el bloc en l'apartat 4.4.3. Cal notar que es converteix el tipus de dada per defecte de Matlab® *double* en *int32*. Aquest canvi és degut a que el programa original en llenguatge C exigeix aquest tipus de dada muntar els paquets enviats posteriorment via port sèrie.

Hi ha un *display* per representar l'estat de la comunicació retornat pel bloc de *sync_write_word*.

Es treuen a la vista general via el port de sortida *commands* les noves posicions dels actuadors.


Vista general
Xdesired
Ydesired
Constant
XY coordinates
orientation
is on plate
Vision
1
Commands
PID(z)
Y axis controller
PID(z)
X axis controller
deg2rad
deg2rad1
ballangle
yrot
Calc
rot_angle2
xrot
position to rotation axis
rad2deg
2nd saturation
ballangle
rotangle
send_Pose
Simulation
rotation axis to instruction
2
new_pose
Commands
execute movement
1.Vision
From Video Device
Grasshopper ...
Y422_320x240
input1
R
G
B
Color Space Conversion
sRGB to L*a*b*
R'
G'
B'
L*
a*
b*
Terminator
Terminator1
>= 20
Threshold
Blob Analysis
BW
Centroid
Count
Compare To Zero
== 0
is on plate
orientation
pixels to real coordinates
pixel_pos
orientation
PosOnPlate
purt_pla
XYcoordinates
2.Execute movement
new_pose
Data Type Conversion
int32
Commands
int32 y1 = dxl_sync_write_word(int32 p3, int32 p4, int32 p5[], int32 u1[])
sync_write_word
Comm. Stat

8. Resultats

Finalment, en aquest apartat es revisen els resultats obtinguts del control de la pilota. És decideix presentar aquests resultats com a millor exponent del treball global realitzat, ja que era l'objectiu últim del projecte. Mitjançant els blocs disponibles de Simulink® per mesurar les variables internes del sistema, s'ha pogut representar el comportament del centroide de la pilota i l'evolució de l'error en ambdós eixos, així com l'evolució de la resposta del controlador en concordança.

En primer lloc, es representa l'evolució en l'espai de la pilota per posicions inicials en les 4 direccions cardinals des del centre de la plataforma. El procediment ha consistit en deixar la pilota estàtica en un punt i tapada per no ser detectada pel sistema de visió. Posteriorment s'ha destapat per deixar actuar el sistema. Tal com es pot apreciar en la figura, la resposta del sistema causa una trajectòria directa cap al centre, on es pot observar una lleugera oscil·lació en la posició final de la bola. Aquest error es considera degut a rugositats en la plataforma final i una concavitat lleugera. També s'ha observat que la pilota no és completament isòtropa en la seva densitat (té uns forats) ni esfèrica, ja que presenta un cert bombament. Aquesta disposició de la massa i la forma, juntament amb el soroll degut a les irregularitats fan entrar sovint al sistema en un estat d'oscil·lació mantinguda . Aquest defecte s'ha trobat per a qualsevol tipus de llançament, trajectòria o punt d'equilibri triat.

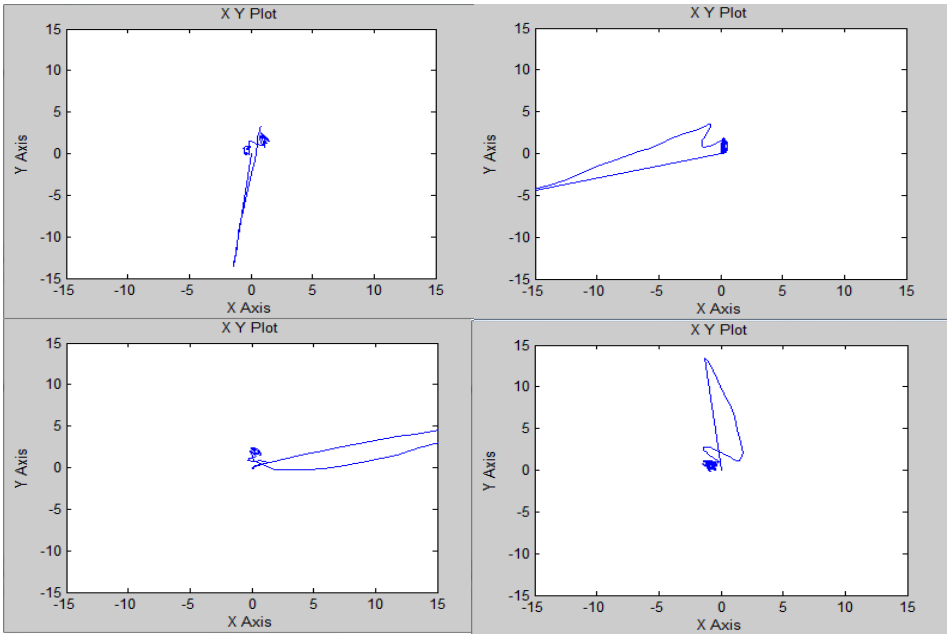


Fig 8.1. *Evolució de la posició per a entrades graó al sistema, col·locant la bola als punts cardinals.*

El temps fins a un error absolut menor d'un centímetre és satisfactori. Pel que fa al temps fins a un error estacionari nul, tot i que finalment s'hi arriba, presenta un caràcter aleatori. Cal esmentar que les marques rectes sobtades inicials tenen a veure amb que la lectura passa de 0 al primer valor mesurat.

A continuació s'han realitzat llançaments de la pilota sobre la plataforma amb velocitats inicials que no tallaven el centre de la plataforma i en ambdues direccions de rotació. També es poden observar llançaments en direcció central on, degut a l'alta velocitat el control no és capaç de frenar la bola abans de passar el centre, i apareix un sobrepuig. En aquest tipus de resposta és un s'aprecia la potència del control PID evitant pertorbacions.

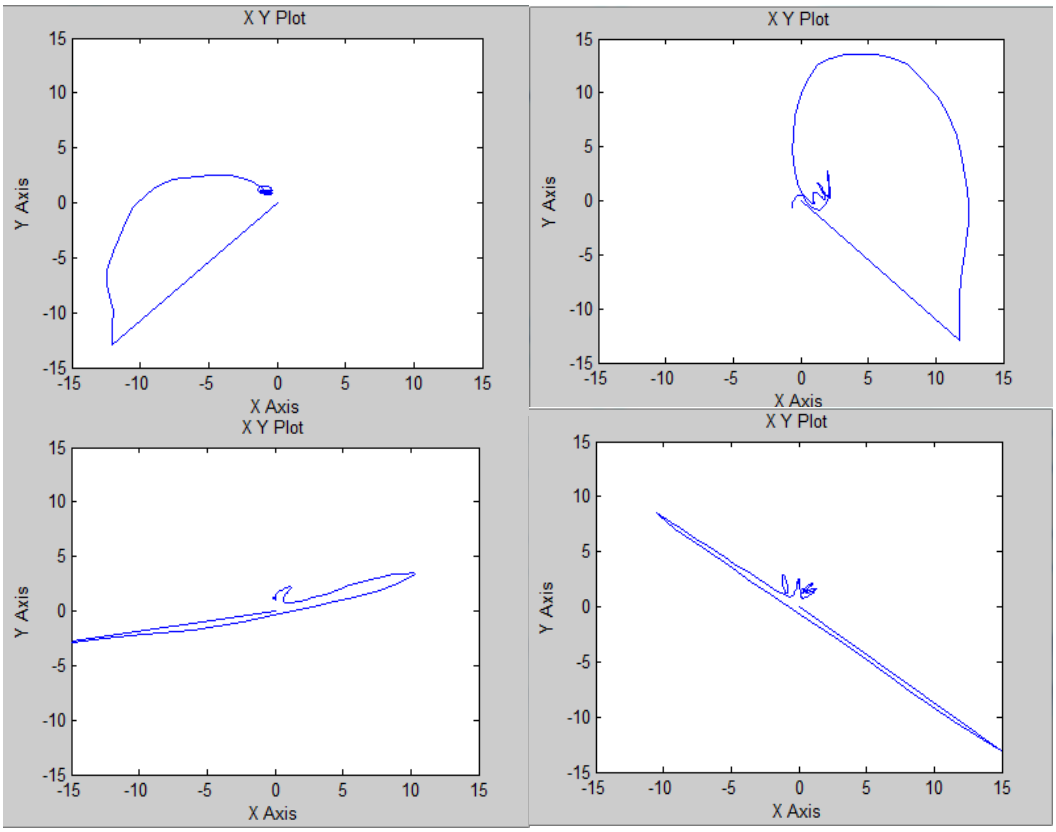


Fig 8.2. *Evolució de la posició per a llançaments amb velocitat inicial des de diferents angles*

Es mostra en la imatge un conjunt de pertorbacions executades seguides manualment en totes direccions des del punt d'equilibri al centre. Al costat es mostra l'evolució de l'error i la resposta del controlador en conseqüència. Es pot veure com s'arriba a l'estat oscil·latori

esmentat anteriorment, d'amplitud variable. En la figura es troben representats els errors de les dues coordenades amb la resposta del controlador.

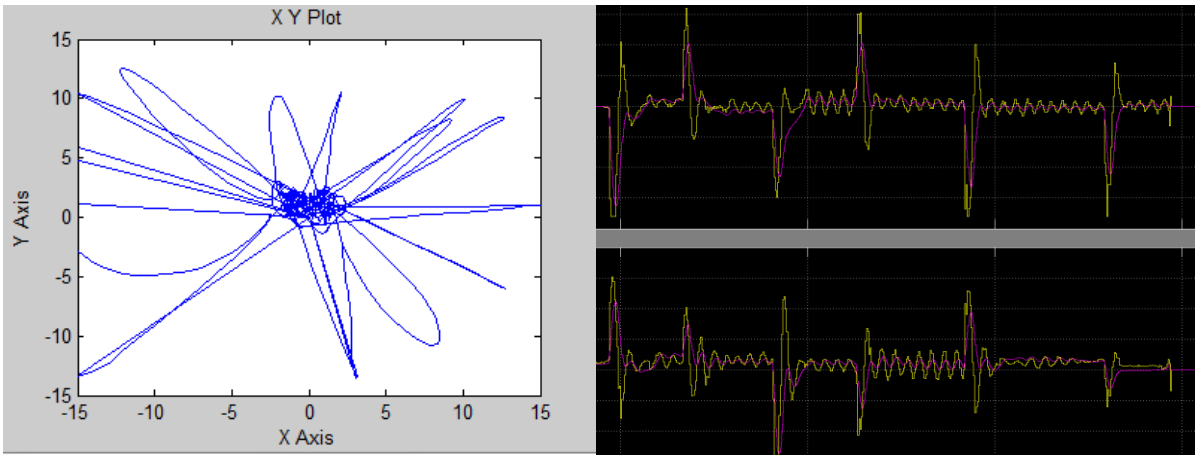


Fig 8.3. *A l'esquerra, evolució de la posició en el pla per diferents pertorbacions. A la dreta, evolució de l'error mesurat en X i Y (lila) i resposta del control en cada eix (groc).*

Posteriorment, s'ha realitzat un conjunt de trajectòries rectilínies de la bola en la plataforma. Els resultats de l'experiment fan apreciable com l'error de posició es fa més gran conforme augmenta la distància al centre.

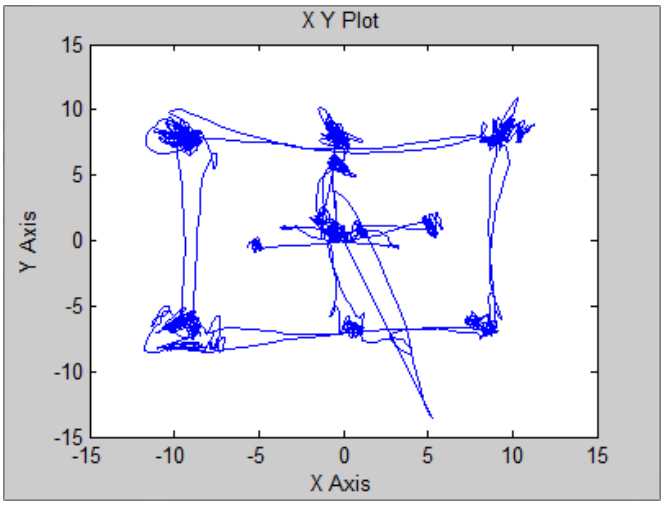


Fig 8.4. *Evolució de la posició en el pla 2D per a diferents* setpoints.

Per acabar, s'han programat diverses trajectòries per la pilota en el pla, amb diferents pertorbacions aplicades durant el moviment. Els resultats mostren com, si bé és possible recórrer aquestes trajectòries, es pot apreciar que el guix del traç final delata un error en el seguiment d'aquesta. Altre cop, s'atribueix a les diferents casuístiques d'orientació de la pilota durant el moviment (ja que no és isòtropa ni rotor esfèric). Aquest error es tradueix en una trajectòria molt difuminada conforme augmenta la complexitat d'aquesta, com en l'última representada.

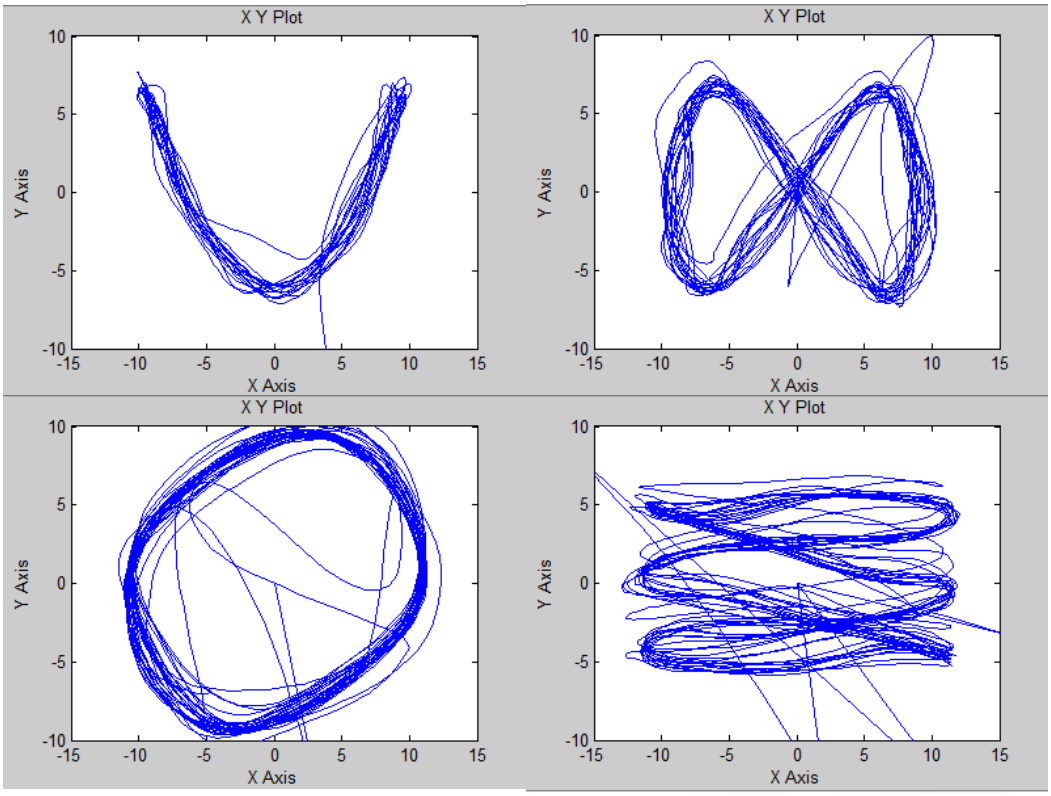


Fig 8.5. *Evolució de la posició al pla per a diferents* trajectòries, amb pertorbacions externes.

Conclusions i treball futur

S'ha construït un robot paral·lel de 6 graus de llibertat amb les especificacions marcades, ja que ha estat capaç de complir amb èxit les tasques de control per les que fou dissenyat. Es conclou per tant que és prou potent per proporcionar un control com el desenvolupat en aquest treball i altres de complexitat superior. Per tal de poder controlar aquest robot, se n'ha resolt la cinemàtica inversa. S'ha particularitzat per a un tipus de moviment en concret i s'ha implementat el resultat en una rutina que genera la seqüència de moviments pels actuadors.

Una millora consistiria en calcular la seqüència de moviments dels actuadors per realitzar rotacions o translacións arbitràries dins l'espai de treball, per poder així augmentar les opcions del control. A més a més, es podria realitzar una calibració prèvia del robot per reduir l'error degut a imprecisions en la seva construcció o en les mesures. També es podria incloure un anàlisi del model dinàmic del robot per incorporar-lo al control.

S'ha resolt amb èxit la comunicació entre la central de procés i els actuadors, ampliant les llibreries de Dynamixel™ 1.0 i creant blocs per a escriptura simultània a la memòria dels actuadors connectats en *daisy chain*. Aquesta modificació pot implementar controls basats en escriptura de posició dels actuadors, velocitats o parells aplicats de manera indiferent.

De la mateixa manera, es podria augmentar la potència dels actuadors per dotar d'una velocitat i acceleració encara superiors. En pujar de gamma d'actuadors, sorgeix l'opció de treballar amb un controlador intern de cadascun configurable i que permetés una millor modelització d'aquests. A més a més, un protocol de comunicació millor podria agilitzar el ritme d'escriptura i lectura en les memòries.

Fent servir Simulink® i Matlab® amb l'ajustament empíric dels controladors, s'ha arribat a resultats satisfactoris tenint en compte l'abast del treball. S'ha obtingut una resposta ràpida com la desitjada amb capacitat de mantenir la pilota equilibrada en un punt, seguir trajectòries de certa complexitat com ara un cercle, un quadrat, o una lemniscata.

En addició, la plataforma és capaç de mantenir la pilota estable en una posició per a pertorbacions externes des del punt d'equilibri, controlar la pilota entrant en la plataforma amb velocitat moderada i independentment de la direcció d'aquesta i controlar la pilota en situacions de bot o de pèrdua de contacte amb la superfície de control. Aquestes característiques tant robustes davant el soroll i les pertorbacions i aproximacions del sistema són degudes a la potència dels controladors PID, que proven ser extremadament versàtils.

El sistema, tot i tenir un error estacionari nul, presenta un temps d'establiment amb una component aleatòria que s'atribueix al soroll degut a les irregularitats de la plataforma i la forma de la pilota entre d'altres. En els assajos realitzats s'observa que l'evolució de la posició de la pilota fins a una zona amb un radi sempre menor a un centímetre d'error de posició pel punt central de la pilota. Aquesta zona d'error s'ha notat que augmenta en funció de la distància al centre. Aquest fet s'atribueix a la linealització realitzada entorn a un punt de treball, ja que s'han emprat mètodes de control lineal.

Per millorar el control realitzat, s'obre un ventall d'opcions diferents tenint en compte que sovint es fan servir els sistemes d'equilibratge com a banc de proves per a nous algorismes de control.

Pel que fa a control no basat en el model es proposa un criteri de zones amb diferents controladors, establint diferents trams de linealització. També es proposa buscar models basats en el *fuzzy control* o amb un controlador i un observador que estimi els paràmetres del control en els diferents estats del sistema.

Per altra banda, amb un model matemàtic precís es podrien avaluar i tractar millor les no linealitats i els acoblaments del sistema MIMO resultant. En aquesta via es podria realitzar una assignació de pols per a un sistema desacoblat i linealitzat, i també definir-lo a trams. També s'obre la via de dissenyar un control no lineal i prosseguir amb investigació en aquesta direcció.

Pel que fa a sistema de visió, s'ha obtingut una binarització ràpida i funcional que extreu la posició de la pilota de manera fiable en tot el camp de visió. És capaç, a més a més, de ser insensible a elements externs com ara la mà de l'operador, tot i que per algunes il·luminacions i tipus de pell pot presentar confusions que alteren la normal identificació de la pilota.

Amb una unitat de procés més potent, es podria realitzar un filtratge de la imatge de major qualitat, amb una major resolució i precisió, que també podrien influir en la qualitat de la resposta ja que el sensat seria més precís.

Finalment cal afegir que, amb una plataforma perfectament plana i adherent, i una pilota perfectament rodona amb una dinàmica més lenta, s'haurien obtingut resultats molt millors però l'objectiu ha estat que el sistema funcionés de forma robusta davant de pertorbacions de tot tipus, fins i tot en les qualitats dels elements constructius.

Agraïments

Vull agrair a en Federico Thomas l'oportunitat donada, els mitjans i l'ajuda, sense els quals no hagués estat possible aquest treball. Gràcies a la seva atenció i confiança, he pogut avançar en un projecte que, tot i que en moments ha semblat que tot eren problemes i que no es veia el final, ha sortit endavant.

També vull agrair a l'Aleix Rull i la gent del laboratori de l'IRI les eines que van posar a la meva disposició i els consells proporcionats en aquesta estada.

Finalment, a la Patri, per escoltar-me, aguantar-me i donar-me el seu suport incondicional.

Bibliografia

Referències bibliogràfiques

[1] Awtar S, Bernard C, Boklund N, Master A, Ueda D, Craig K (2002) Mechatronic Design of a Ball on Plate Balancing System. Mechatronics. Vol. 12, No. 2, pp. 217-228.

[2] Basic input output elements. (2011) *Ball on plate*. Recuperat des de http://bioe.sourceforge.net/contributions.php

[3] Fan X, Zhang N, Teng S (2003) Trajectory Planning and Tracking of Ball and Plate System Using Hierarchical Fuzzy Control Scheme. Fuzzy Sets and Systems. Vol. 144, No. 2, pp. 297-312.

[4] Park J H, Lee Y J (2003)Robust Visual Servoing for Motion Control of the Ball on a Plate. Mechatronics. Vol. 13, No. 7, pp. 723-738.

[5] Rad A B, Chan P T, Lo W L, Mok C K (2003) An Online Learning Fuzzy Controller. IEEE Transactions on Industrial Electronics. Vol. 50, No. 5, pp. 1016-1021.

[6] Yip P T (2004) Symbol-based Control of a Ball on Plate Mechanical System. Master's Thesis.University of Maryland.

[7] Moarref M, Saadat M, Vossoughi G (2008) Mechatronic Design and Position Control of a Novel Ball and Plate System. Proceedings of 16th Mediterranean Conference on Control and Automation, pp. 1071-1076.

[8] Liu D, Tian Y, Duan H (2009)Ball and Plate Control System Based on Sliding Mode Control with Uncertain Items Observe Compensation. Proceedings of IEEE International Conference onIntelligent Computing and Intelligent Systems, pp. 216-221.

[9] Moreno-Armendáriz M A,Pérez-Olvera C A,Rodriguez F O, Rubio E (2010)Indirect Hierarchical FCMAC Control for the Ball and Plate System. Neurocomputing, Vol. 73, No. 13-15, pp. 2454-2463.

[10] Moreno-Armendáriz M A, Rubio E, Pérez-Olvera C A (2010) Design and Implementation of a Visual Fuzzy Control in FPGA for the Ball and Plate System. International Conference on Reconfigurable Computing, pp. 85-90.

[11] Yuan D, Zhang Z (2010)Modeling and Control Scheme of the Ball-Plate

Trajectory-Tracking Pneumatic System with a Touch Screen and a Rotary Cylinder. IET Control Theory and Applications, Vol. 4, No. 4, pp. 573-589.

[12] Hutchinson S, Hager GD, Corke PI (1996) A Tutorial on Visual Servo Control.IEEE Transactions on Robotics and Automation, Vol. 12, No. 5, pp. 651-670.

[13] Batlle J, Marti J, Ridao P, Amat J (2002) A New FPGA/DSP-based Parallel Architecture for Real-Time Image Processing. Real-Time Imaging, Vol. 8, No. 5, pp. 345-356.

[14] Torres-Huitzil C, Arias-Estrada M (2004) Real-Time Image Processing with a Compact FPGA-based Systolic Architecture. Real-Time Imaging, Vol. 10, No. 3, pp. 177-187.

[15] Martin J L, Zuloaga A, Cuadrado C, Lazaro J, Bidarte U (2005) Hardware Implementation of Optical Flow Constraint Equation Using FPGAs. Computer Vision and Image Understanding, Vol. 98, No. 3, pp. 462-490.

[16] Diaz J, Ros E, Pelayo F, Ortigosa E M, Mota S (2006) FPGA-based Real-Time Optical-Flow System. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, No. 2, pp. 274-279.

[17] Full Motion Dynamics. (2012) *Ball on Plate PID control with 6DoF Stewart platform.* Recuperat des de: http://www.fullmotiondynamics.com/

[18] Robotis E-Manual. (2015) *Dynamixel Ax-12+. Recuperat des de* http://support.robotis.com/en/product/dxl_main.htm

[19] "Triangulo HSV" by Samus_ - Treball propi. Licensed under CC BY-SA 3.0 via WikimediaCommonshttps://commons.wikimedia.org/wiki/File:Triangulo_HSV.png#/media/File:Triangulo_HSV.png

[20] Lab color at luminance 25 percent" by The original uploader was JakobVoss de la Viquipèdia en alemany - Originally from en.wikipedia via de.wikipedia; description page is/was here.. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Lab_color_at_luminance_25_percent.png#/media/File:Lab_color_at_luminance_25_percent.png

[21] Wisama Khalil, Ouarda Ibrahim. General Solution for the Dynamic Modeling of Parallel Robots. Journal of Intelligent and Robotic Systems, Springer Verlag (Germany), 2007, 49, pp. 19-37.

[22] J.-P. Mertlet. (2006) Parallel robots 2nd Edition Ed. Springer pp. 131-132

[23] N. Rojas, J. Borràs, and F. Thomas, "The Octahedral Manipulator Revisited," Proceedings of the 2012 IEEE International Conference on Robotics and Automation, May 14-18, Saint Paul, Minnesota, USA.

[24] P.I. Corke, "Robotics, Vision & Control", Springer 2011

[25] "PID en updated feedback" by TravTigerEE - Own work. Licensed under CC BY-SA 3.0 viaWikimediaCommonshttps://commons.wikimedia.org/wiki/File:PID_en_updated_feedback.svg#/media/File:PID_en_updated_feedback.svg

[26] Rule-Based Autotuning Based on Frequency Domain Identification," Anthony S. McCormack and Keith R. Godfrey, IEEE Transactions on Control Systems Technology, vol 6 no 1, January 1998.