



UNIVERSIDADE DA CORUÑA

ESCOLA POLITÉCNICA SUPERIOR

Máster en Enxeñaría Industrial

TRABALLO DE FIN DE MÁSTER

TFM Nº: **10**

TÍTULO: **DESENVOLVEMENTO EN ROS DE LIBRERÍAS
DE COMPORTAMENTO BÁSICO PARA O ROBOT NAO**

AUTOR: **ALEJANDRO ROMERO MONTERO**

TUTOR: **FRANCISCO JAVIER BELLAS BOUZA
PEDRO TRUEBA MARTÍNEZ**

DATA: **FEBREIRO DE 2017**

Fdo.: O AUTOR

Fdo.: OS TITORES

TÍTULO: **DESENVOLVEMENTO EN ROS DE LIBRERÍAS**

DE COMPORTAMENTO BÁSICO PARA O ROBOT NAO

ÍNDICE XERAL

PETICIONARIO: **ESCOLA POLITÉCNICA SUPERIOR**

CAMPUS DE ESTEIRO, RÚA MENDIZÁBAL, S/N

15403 - FERROL

DATA: **FEBREIRO DE 2017**

AUTOR: **O ALUMNO**

Fdo.: **ALEJANDRO ROMERO MONTERO**

I	ÍNDICE XERAL	3
	Contidos do TFM	5
	Índice de figuras	7
	Índice de táboas	9
II	MEMORIA	11
	Índice do documento Memoria	13
1	Introducción	15
2	Obxectivos	17
3	Fundamentos tecnolóxicos	19
3.1	Nao	19
3.2	ROS	21
3.2.1	Introducción	21
3.2.2	Características	22
3.2.3	Rede de comunicación en ROS	23
3.3	Python	25
3.4	Módulos e librerías adicionais	25
3.4.1	OpenCV	26
3.4.2	APIs e módulos NAOqi	26
4	Antecedentes	29
4.1	Introducción	29
4.2	Control a través de supervisión humana de robots para tarefas industriais rutineiras	30
4.3	Teleoperación / telerobótica en ambientes perigosos ou inaccesibles	32
4.4	Vehículos autónomos, ferroviarios e aeronaves comerciais	34
4.5	Interacción social humano-robot	35
4.5.1	HRI en robots terapéuticos	36
4.5.2	HRI en robots cirúrxicos	37
4.5.3	HRI en robots de entretenimento	38
4.5.4	HRI en robots educativos	39
4.5.5	HRI con Nao. Uso académico e científico	40
4.6	Conclusión	42
5	Requisitos de deseño	43
6	Desenvolvemento das solucións	45
6.1	Arquitectura global	45
6.2	Liberaría de detección de imaxe	47
6.2.1	Detección de obxectos de cor	48
6.2.2	Detección de caras	55
6.3	Liberaría de son	60
6.3.1	Recoñecemento de voz	60
6.3.2	Sintetizador de voz	61
6.3.3	Localización da fonte de son	61

6.4 Librería de movementos básicos	64
6.4.1 Movemento do corpo	64
6.4.2 Movemento de agarre de obxectos	73
7 Probas e resultados	83
7.1 Contorna das probas	83
7.2 Seguimento dun obxecto de cor coa cabeza ou con todo o corpo	84
7.3 Seguimento dunha cara coa cabeza	88
7.4 Seguimento e agarre dun obxecto de cor	90
7.5 Orientación cara a unha fonte de audio e recoñecemento da fala	94
7.6 Exemplo global de interacción con humanos	96
8 Conclusíons e traballo futuro	101
8.1 Traballo futuro	102

Índice de figuras

3.1.0.1	Robot humanoide Nao	20
3.2.3.1	Comunicación entre dous nodos en ROS	24
3.2.3.2	Exemplo de comunicación entre dispositivos baseada en ROS	24
3.3.0.1	Exemplo de programación en ROS a través de Python	25
3.4.1.1	Comunicación entre OpenCV e ROS por medio da librería CvBridge	26
4.2.0.1	Robot Baxter de Rethink Robotics	32
4.3.0.1	Robots participantes no DARPA Robotics Challenge	33
4.5.1.1	Robot Romeo de Aldebaran Robotics	36
4.5.2.1	Sistema quirúrxico Da Vinci	37
4.5.3.1	Robots Aibo de Sony	39
4.5.4.1	Robots creados co kit Mindstorms de Lego	40
4.5.5.1	Uso de Nao como robot terapéutico	41
6.1.0.1	Compoñentes involucrados no funcionamento das diferentes librerías	45
6.1.0.2	Esquema de comunicación Nao-Python-ROS	46
6.2.0.1	Posicionamento e campo de visión das cámaras situadas na cabeza de Nao	47
6.2.1.1	Uso da función 'HoughCircles' para o recoñecemento da pelota	49
6.2.1.2	Rectángulo representativo dos píxeles utilizados para a estimación	50
6.2.1.3	Diagrama do espazo de cor HSV	51
6.2.1.4	Exemplo de aplicación de máscara e posterior procura do contorno	53
6.2.1.5	Exemplo de detección cando existen varias bolas da mesma cor	54
6.2.1.6	Exemplo de detección cando existen varias bolas da diferentes cores	54
6.2.1.7	Exemplo de detección con obxectos da mesma cor na imaxe	54
6.2.1.8	Exemplo da imaxe publicada para realimentación ao usuario	55
6.2.2.1	Funcionamento dun detector baseado en multiples etapas ou fervenzas . . .	56
6.2.2.2	Exemplo de detección de caras	57
6.2.2.3	Influencia do factor de escala no algoritmo de recoñecemento facial	58
6.2.2.4	Exemplo de detección cando se ven varias caras na imaxe	59
6.3.3.1	Dependencia entre a posición da fonte de son e as distancias que a onda sonora percorre para alcanzar os micrófonos de Nao	62
6.4.1.1	Marcos de referencia Torso e Robot	65
6.4.1.2	Marco de referencia World	65
6.4.1.3	Convención de signos (Roll, Pitch, Yaw)	66

6.4.1.4	Articulacións da cabeza	67
6.4.1.5	Articulacións do brazo esquierdo	68
6.4.1.6	Articulacións da pelvis	68
6.4.1.7	Articulacións da perna esquerda	69
6.4.1.8	Posición dos distintos efectores do robot Nao	71
6.4.1.9	Localización dos diferentes motores de Nao	71
6.4.1.10	Comunicación vía ROS para a librería de desprazamento	72
6.4.1.11	Comunicación vía ROS para as funcións de movemento de articulacións	73
6.4.2.1	Exemplos de posicionamiento do centro da pelota respecto á franxa central da imaxe	77
6.4.2.2	Exemplos de aliñación da cabeza respecto ao corpo	77
6.4.2.3	Secuencia de agarre de obxectos	78
6.4.2.4	Coordenadas X e Y respecto ao marco de referencia do torso do robot	79
6.4.2.5	Altura da pelota en función do ángulo entre a cabeza e a horizontal	80
6.4.2.6	Gráfico coa altura da pelota respecto ao torso do robot e o valor de HeadPitch	81
6.4.2.7	Exemplo de agarres a diferentes alturas	82
7.1.0.1	Elementos para a realización das diferentes probas	83
7.2.0.1	Botóns táctiles situados na cabeza de Nao	84
7.2.0.2	Detección da forma circular da pelota na posición de recoñecemento	85
7.2.0.3	Detección dos píxeles más representativos para a estimación	85
7.2.0.4	Modo de funcionamento dos botóns táctiles	86
7.2.0.5	Exemplo do código empregado para calcular o movemento da cabeza e a súa restriccción	86
7.2.0.6	Diagrama de fluxo para o exemplo de seguimento dun obxecto de cor	87
7.3.0.1	Exemplo de detección cando se ven varias caras na imaxe	89
7.3.0.2	Diagrama de fluxo para o exemplo de seguimento dunha cara coa cabeza	89
7.4.0.1	Posición de Nao con corpo e cabeza aliñados	91
7.4.0.2	Exemplo do código empregado para a implementación do regulador proporcional	92
7.4.0.3	Diagrama de bloques xenérico do regulador empregado	92
7.4.0.4	Fluxograma para o exemplo de seguimento e agarre dun obxecto	93
7.4.0.5	Exemplo da secuencia de agarre de obxectos	94
7.5.0.1	Diagrama de fluxo para o exemplo global de son	96
7.6.0.1	Escenario de partido do exemplo global de interacción	97
7.6.0.2	Nao virando a súa cabeza cara a fonte do son	98
7.6.0.3	Imaxe do que está a ver Nao cando se detecta a bóla	99
7.6.0.4	Nao realiza o agarre da pelota que se atopa nas mans do usuario	99
7.6.0.5	Fluxograma do exemplo global de interacción con humanos	100

Índice de táboas

3.1.0.1 Táboa de características do robot humanoide Nao	21
6.2.1.1 Límites HSV medios para diferentes cores	51
6.3.3.1 Táboa cos modos de seguimento	63
6.3.3.2 Táboa cos posibles obxectivos de seguimento	63
6.4.1.1 Rango de movemento das articulacións da cabeza	67
6.4.1.2 Rango de movemento das articulacións do brazo esquerdo	69
6.4.1.3 Rango de movemento das articulacións da pelvis	69
6.4.1.4 Rango de movemento das articulacións da perna esquerda	70
6.4.1.5 Articulacións pertencentes a cada cadea	70
6.4.2.1 Valores de HeadPitch en función da altura á que está situado o obxecto	81

TÍTULO: **DESENVOLVEMENTO EN ROS DE LIBRERÍAS**

DE COMPORTAMENTO BÁSICO PARA O ROBOT NAO

MEMORIA

PETICIONARIO: **ESCOLA POLITÉCNICA SUPERIOR**

CAPUS DE ESTEIRO, RÚA MENDIZÁBAL, S/N

15403 - FERROL

DATA: **FEBREIRO DE 2017**

AUTOR: **O ALUMNO**

Fdo.: **ALEJANDRO ROMERO MONTERO**

Índice do documento MEMORIA

1 Introducción	15
2 Obxectivos	17
3 Fundamentos tecnolóxicos	19
3.1 Nao	19
3.2 ROS	21
3.2.1 Introdución	21
3.2.2 Características	22
3.2.3 Rede de comunicación en ROS	23
3.3 Python	25
3.4 Módulos e librerías adicionais	25
3.4.1 OpenCV	26
3.4.2 APIs e módulos NAOqi	26
4 Antecedentes	29
4.1 Introdución	29
4.2 Control a través de supervisión humana de robots para tarefas industriais rutineiras	30
4.3 Teleoperación / telerobótica en ambientes perigosos ou inaccesibles	32
4.4 Vehículos autónomos, ferroviarios e aeronaves comerciais	34
4.5 Interacción social humano-robot	35
4.5.1 HRI en robots terapéuticos	36
4.5.2 HRI en robots cirúrxicos	37
4.5.3 HRI en robots de entretemento	38
4.5.4 HRI en robots educativos	39
4.5.5 HRI con Nao. Uso académico e científico	40
4.6 Conclusión	42
5 Requisitos de deseño	43
6 Desenvolvemento das solucións	45
6.1 Arquitectura global	45
6.2 Librería de detección de imaxe	47
6.2.1 Detección de obxectos de cor	48
6.2.1.1 Detección de cor	48
6.2.1.2 Espazo de cor HSV	51
6.2.1.3 Detección de formas circulares	52
6.2.2 Detección de caras	55
6.3 Librerías de son	60
6.3.1 Recoñecemento de voz	60

6.3.2 Sintetizador de voz	61
6.3.3 Localización da fonte de son	61
6.4 Librería de movementos básicos	64
6.4.1 Movemento do corpo	64
6.4.1.1 Marcos de referencia (Frames)	64
6.4.1.2 Articulacións, efectores e cadeas	66
6.4.1.2.1 Convención de signos	66
6.4.1.2.2 Articulacións da cabeza	66
6.4.1.2.3 Articulacións dos brazos	67
6.4.1.2.4 Articulacións da pelvis	67
6.4.1.2.5 Articulacións das pernas	69
6.4.1.2.6 Cadeas	70
6.4.1.2.7 Efectores	70
6.4.1.3 Motores	71
6.4.1.4 Desprazamento do robot	72
6.4.1.5 Movemento das articulacións	73
6.4.2 Movemento de agarre de obxectos	73
6.4.2.1 Cinemática inversa a través do módulo “Cartesian Control”	75
6.4.2.2 Agarre de obxectos a diferentes alturas: Pasos a seguir	76
6.4.2.3 Localización do obxecto no espazo 3D	78
6.4.2.3.1 Coordenada X	78
6.4.2.3.2 Coordenada Y	79
6.4.2.3.3 Coordenada Z	80
7 Probas e resultados	83
7.1 Contorna das probas	83
7.2 Seguimento dun obxecto de cor coa cabeza ou con todo o corpo	84
7.3 Seguimento dunha cara coa cabeza	88
7.4 Seguimento e agarre dun obxecto de cor	90
7.5 Orientación cara a unha fonte de audio e recoñecemento da fala	94
7.6 Exemplo global de interacción con humanos	96
8 Conclusións e traballo futuro	101
8.1 Traballo futuro	102

1 Introducción

Este Traballo Fin de Máster enmárcase dentro do proxecto europeo de investigación DREAM (Deferred Restructuring of Experience in Autonomous Machines) [1] que se leva a cabo no Grupo Integrado de Enxeñaría (GII) e que está financiado pola Unión Europea. O obxectivo global do devandito proxecto consiste no desenvolvemento dunha arquitectura cognitiva que permita aos dispositivos robóticos realizar tarefas de aprendizaxe e de mellora do que aprenderon. Estas tarefas realizaranse mentres os dispositivos se atopen “durmidos”, en analoxía ao funcionamento da aprendizaxe nos seres humanos durante a fase de sono, de forma que consoiden así o coñecemento adquirido durante a fase de operación en “tempo de vida”.

Para levar a cabo os exemplos prácticos dentro do proxecto DREAM, adquiríronse dúas unidades do robot humanoide Nao. Este robot vén de serie cun conxunto de comportamentos básicos de recoñecemento de obxectos e movemento, pero non son modificables e unicamente están dispoñibles no seu entorno de programación nativo. Debido a isto establecécese a necesidade de desenvolver unhas librerías con comportamentos más amplos que permitan aos investigadores do GII centrarse na aprendizaxe a partir da interacción con humanos, un dos principais problemas a solucionar no proxecto DREAM.

Dentro deste proxecto estableceuse o requisito de que todos os desenvolvimentos se realicen mediante os estándares de ROS (Robot Operating System), polo que é necesario implementar os devanditos comportamentos básicos neste novo sistema. Polo tanto, re-implementaranse comportamentos de baixo nivel (navegación básica, localización, mapeado) e desenvolveranse outros novos de máis alto nivel (seguir un obxecto de cor, coler un obxecto, atender a sons, etc) que serán utilizados no proxecto DREAM como coñecemento innato que o robot ten á súa disposición.

Estrutura da memoria

A memoria do traballo está estruturada en oito capítulos, sendo o primeiro deles esta **Introdución**.

- Segundo capítulo: **Obxectivos**. Neste capítulo indicaranse o obxectivo e sub-obxectivos principais do TFM e a súa xustificación.
- Terceiro capítulo: **Fundamentos tecnolóxicos**. Fálase sobre o robot humanoide Nao, o sistema operativo para robots (ROS) e a linguaxe de programación Python empregados para o desenvolvemento das diferentes librerías que componen este traballo, así como o resto de módulos ou librerías adicionais utilizadas.
- Cuarto capítulo: **Antecedentes**. Todos aqueles aspectos necesarios para a comprensión da importancia da interacción humano-robot e os diferentes modos de interacción existentes, acabando polo estudo da solución final adoptada.
- Quinto capítulo: **Requisitos de deseño**. Neste capítulo describiranse as bases e datos de partida establecidos polos investigadores do GII que condicionan as solucións técnicas do mesmo.
- Sexto capítulo: **Desenvolvemento das solucións**. Elabórase o desenvolvemento técnico do traballo, o modelo conceptual, a metodoloxía seguida e os detalles de implementación do mesmo.
- Séptimo capítulo: **Probas e resultados**. Coméntanse os resultados do desenvolvemento das solucións e expóñense as aplicacións de exemplo desenvolvidas utilizando as diferentes librerías de interacción nos robots Nao.
- Oitavo capítulo: **Conclusíons e traballo futuro**. Neste derradeiro capítulo coméntase o resultado xeral do traballo así como o traballo futuro a desenvolver.

2 Obxectivos

O obxectivo principal deste Traballo Fin de Máster é levar a cabo o “desenvolvemento de librerías en ROS que implementen unha serie de comportamentos básicos no robot autónomo Nao”. Ditas librerías deberán de ser probadas sobre o robot real e a súa programación efectuarase en Python.

As librerías a realizar deberán permitir o desenvolvemento de comportamentos básicos de interacción con humanos. En consecuencia, establecense como sub-obxectivos os seguinte desenvolvementos:

- Librería de detección de imaxe, de forma que sexa posible a detección de propiedades básicas da imaxe (como cor ou forma), ou outras más avanzadas como as caras.
- Librería de son, para facer máis proveitosa a interacción ten que ser posible tanto o recoñecemento de voz como a súa sintetización, ademais de poder localizar a procedencia da fonte de son.
- Librería de movementos básicos, que permita o movemento do corpo do robot para relacionarse coa contorna que o rodea. Deberán ser posibles accións como camiñar ou o agarre de obxectos situados en lugares determinados.

Finalmente, realizaranxe unha serie de exemplos de aplicación para deixar claro o seu funcionamento e demostrar que se poden utilizar todas as librerías de forma conjunta.

3 Fundamentos tecnolóxicos

Neste capítulo introdúcese a base teórica sobre a que se desenvolve o presente Traballo de Fin de Máster, en concreto falarase do robot humanoide Nao, do sistema operativo para robots (ROS), da linguaxe de programación Python, e do resto de módulos ou librerías adicionais utilizadas.

3.1. Nao

Nao [2] é un robot humanoide autónomo e programable desenvolvido por Aldebaran Robotics, unha compañía de robótica francesa establecida en París. É froito dunha combinación de enxeñaría mecánica e software e está composto por multitud de sensores, motores e un software pilotado por un sistema operativo a medida baseado en Linux: NAOqi OS.

Desde o seu lanzamento en 2008 foron creadas diversas versións. A Edición Académica de Nao foi desenvolvida para universidades e laboratorios para propósitos de investigación e educación. Foi posto a disposición das institucións en 2008, pero non estaría a disposición do público ata 2011. Varias melloras da plataforma Nao foron desenvolvidas, incluíndo o Nao Next Gen en 2011 e o Nao Evolution en 2014. A versión actual de Nao é a V5, mostrado na figura 3.1.0.1, que consta dunha serie de capacidades que serven para crear unha interacción natural con usuarios humanos:

- **Movemento:** 25 graos de liberdade repartidos entre cabeza, brazos, tronco e pernas, ademais de contar con mans articuladas que, entre outras cousas, permiten o agarre de obxectos ou a interacción con humanos, a pesar das limitacións que presentan, xa que non hai sensores táctiles nin de forza nelas que permitan coñecer de forma rápida se se está a agarrar algúun obxecto. Isto, xunto coa súa forma humanoide permítelle moverse e adaptarse ao mundo que o rodea. A súa unidade de inercia permítelle manter o equilibrio e coñecer se se atopa de pé ou tirado.
- **Sensorización:** Os numerosos sensores de contacto ou táctiles situados na súa cabeza, mans e pés, así como os seus sonars e a placa de inercia, permítenlle percibir o ambiente que o rodea e coñecer a súa orientación. Nao dispón de 3 sensores táctiles na cabeza que se poden usar como realimentación do usuario ao robot en algúun tipo de interacción; un botón no peito para coñecer o seu estado; 3 sensores capacitivos en cada man que,

por exemplo, se poden usar para saber se alguén lhas está agarrando; e máis un bumper na punta de cada pé para coñecer se está a chocar con algo.

- **Oído e fala:** Con 4 micrófonos direccionais (para recoñecemento de voz e localización de son) e 2 altofalantes (para a síntese de texto a voz multilingüe), Nao pode interactuar cos humanos dunha maneira completamente natural, escouitando e falando.
- **Vista:** Nao está equipado con dúas cámaras que lle permiten observar o seu ambiente en alta resolución , axudándoo así a recoñecer formas, caras e obxectos. O gran problema destas cámaras é que non permiten recoñecer a profundidade, polo que para tales fins soen utilizarse cámaras 3D externas montadas sobre a cabeza do propio robot, como por exemplo a Kinect.
- **Comunicación:** Nao é capaz de usar un rango diferente de modos de conexión (WiFi, Ethernet) que lle permiten comunicarse con un ordenador persoal, con outros robots ou con calquera outro dispositivo.
- **Pensamento:** A parte máis importante coa que conta Nao é o seu procesador: conta cunha CPU Intel Atom a 1.6 GHz, 1 Gb de RAM, 2 Gb de memoria flash e 8 GB Micro SDHC. Estas características son as que permiten realizar a maioría das operacións dentro do propio robot, pero cando os cálculos teñen gran complexidade e este procesador se ve sobrepasado, débense facer a través dun ordenador exterior, o cal é posible grazas aos modos de conexión comentados con anterioridade.

Estas características e algunas outras móstranse de forma resumida na táboa [3.1.0.1](#).



Figura 3.1.0.1 – Robot humanoide Nao

O robot tamén vén cun software propio que inclúe unha ferramenta de programación gráfica chamada Choregraphe, un paquete de software de simulación e un kit de desenvolvemento

(IDE) de software. Nao é ademais compatible con Microsoft Robotics Studio, Cyberbotics Webots, e o Estudio Gostai URBI.

No que a autonomía se refire, Nao atópase limitado a uns 30 minutos de operación, por tanto para aplicacións que requiran largos tempos de funcionamento pódese facer necesario conectalo á rede para que cargue ou mesmo incluír tempos de descanso para evitar o sobre-quentamento dos seus motores.

Especificacións	
Modelo	Nao V5 Evolution (2014)
Altura	58 centímetros
Peso	4.3 kg
Fonte de alimentación	Batería de litio de 46 W · h
Autonomía	90 minutos (uso continuo)
Graos de liberdade	25
CPU	Intel Atom @ 1.6 GHz
Sistema Operativo interno	NAOqi 2.0 (baseado en Linux)
Sistemas Operativos compatibles	Windows, Mac OS, Linux
Linguaxes de programación	C++, Python, Java, MATLAB, Urbi, C, .Net
Sensores	2 cámaras HD, 4 micrófonos, telémetro sónar, 2 emisores e receptores infravermellos, placa de inercia, 9 sensores táctiles, 8 sensores de presión
Conectividade	Ethernet, WiFi

Táboa 3.1.0.1 – Táboa de características do robot humanoide Nao

3.2. ROS

3.2.1. Introdución

ROS [3] (do inglés Robot Operating System) é un framework para o desenvolvemento de software de robótica. É unha colección de ferramentas, bibliotecas e convencións que teñen como obxectivo simplificar a tarefa de crear un comportamento robótico complexo e robusto a través dunha ampla variedade de plataformas robóticas.

O software está estruturado como un gran número de pequenos programas (nodos) que pasan rapidamente mensaxes entre si. Este paradigma foi elixido para fomentar a reutilización do software de robótica fóra do robot e contorna particular no que foi creado. De feito, esta estrutura lixeiramente axustada permite a creación de módulos xenéricos que son aplicables a amplas clases de hardware robótico e de software, facilitando o uso compartido de códigos e a reutilización entre a comunidade robótica global.

ROS non é un sistema operativo e debe executarse sobre plataformas baseadas en UNIX. O software de ROS utilízase fundamentalmente en Ubuntu ánda que os diversos usuarios

tamén estiveron realizando contribucións noutras plataformas Linux como Fedora ou Gentoo, así como en Mac OS ou incluso en Microsoft Windows, aínda que esta última opción non foi moi explorada.

O repositorio de software de ROS contén multitud de paquetes que proporcionan funcionalidades de forma simple, de maneira que o programador poida centrarse no desenvolvemento do control do seu robot sen ter que preocuparse de implementar os algoritmos e técnicas más comúns.

3.2.2. Características

As principais características de ROS e quizais tamén os seus puntos fortes atópanse en:

- **Computación distribuída:** a maioría dos robots actuais requieren de software que executan diferentes procesos en diferentes ordenadores. Tamén é unha práctica común dividir o software do robot en pequenas partes independentes que cooperan para lograr un obxectivo proposto. Enténdese pois que é necesaria unha canle de comunicación entre os múltiples procesos que están a ter lugar en diferentes ou incluso nun mesmo ordenador. ROS proporciona ferramentas que fan moi sinxelo implementar este tipo de comunicación nos programas.

- **Reutilización de software:** o vertiginoso progreso que experimentou o campo da robótica nos últimos anos deu como resultado un bo número de algoritmos para resolver algúns tarefas típicas como a navegación, planificación de tarefas, creación de mapas, etc. Está claro que a existencia destes algoritmos só é útil se hai algún modo de aplicalos a novos contextos sen necesidade de reimplementalos de maneira particular para cada caso. ROS proporciona funcionalidades que permiten evitar este tipo de problemas:

1. Incorpora paquetes onde xa veñen implementadas e probadas versións estables de moitos algoritmos importantes en robótica.
2. Non impón restricións ao código desenvolvido noutros frameworks.
3. Independencia da linguaxe de programación utilizada: ROS está actualmente implementado en Python, C++ e Lisp, existindo librerías experimentais en Java e Lua.

- **Facilidade para a realización de probas:** unha das razóns polas que o desenvolvemento de software para robots é a miúdo máis difícil que outros tipos de desenvolvemento é que a realización de probas consume moito tempo e é frecuente que aparezan erros, por non falar de que en moitos casos non se dispón dun robot físico no que realizar as probas. ROS proporciona diferentes alternativas de abordar o problema:

1. ROS separa os sistemas de baixo nivel encargados do control do hardware dos de alto nivel, enfocado más ben ao procesado e á toma de decisións. Grazas a

esta separación, pódense substituir estes programas de baixo nivel e o software correspondente por un simulador, co que permite centrarse en probar a parte de alto nivel do sistema.

2. ROS tamén proporciona unha forma sinxela de gravar e volver reproducir datos dos sensores ou outro tipo de mensaxes. Isto permite aproveitar máis eficientemente o tempo investido na realización de probas que se se traballase directamente tomando medidas do robot físico.

- **Popularidade:** posiblemente o principal punto forte de ROS sexa a gran cantidade de persoas do ámbito da robótica que o utilizan, o que implica que ten unha ampla “masa crítica” que se encarga de actualizar constantemente os repositorios, poñer a disposición de calquera persoa novos algoritmos ou funcionalidades froito dalgún proceso de investigación, corrixir calquera malfuncionalidade que puidese xurdir na plataforma, actualización constante da mesma para que sexa posible traballar co hardware máis punteiro, etc.

3.2.3. Rede de comunicación en ROS

A wiki de ROS [4] define o seu “Grafo Computacional” como a rede de pares entre os procesos ROS que procesan os datos xuntos. Con todo é necesario puntualizar que existe un proceso mestre “Master”, que centralizará as conexións, e será dentro do espazo provisto por este onde terá lugar a rede de pares. Esta rede está composta polos seguintes elementos:

- **Nodos:** procesos que realizan unha computación. Dado que ROS está deseñado para ofrecer un control moi fino, un sistema de control robótico adoita abarcar varios nodos. Un nodo escríbese utilizando unha librería de cliente de ROS como rosp y roscpp.
- **Master:** prové servizos de rexistro de nomes e consulta ao resto do grafo computacional, funciona de punto de encontro entre varios nodos.
- **Servidor de parámetros:** permite almacenar datos baixo un índice nunha localización central, é parte do Master.
- **Mensaxes:** a forma de comunicación entre nodos, trátase dunha estrutura de datos composta recursivamente por estruturas (similares aos “struct” de C), listas e tipos de datos primitivos.
- **Temas (Topics):** estrutura similar a un bus de mensaxes, permite a publicación de mensaxes e a subscrición a el para recibilos como se mostra na figura 3.2.3.1. Ao non ser necesaria a interacción entre os nodos que publican e os subscritos á canle, mellora a interoperabilidade e a capacidade de extensión de ambos.
- **Servizos:** mecanismo que permite a un nodo ofrecer un punto onde realizar unha petición e dar unha resposta a esa petición concreta. Isto pode ser implementado usando

temas, pero os servizos simplifican esta interacción. Na figura 3.2.3.1 móstrase a diferenza entre un servizo e un tema á hora da comunicación entre dous nodos.

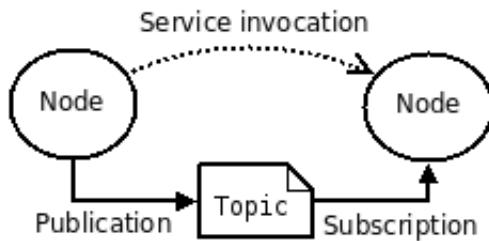


Figura 3.2.3.1 – Comunicación entre dous nodos en ROS

- **Bolsas (bags):** formato para almacenar e reproducir mensaxes de ROS. Permite a reutilización de datos difíciles de recoller pero útiles para desenvolvemento e probas, como información de sensorización.

En resumo, a comunicación que ofrece ROS baséase na publicación e a subscrición a temas, onde un tema é simplemente un colector de datos cun nome que garda só os datos entrantes e empuxa as actualizacións de estado a todos os nodos subscritores. Un nodo é un cliente nunha rede ROS e pode ser un subscritor, un editor ou ambos. Cada nodo ten que ser identificado por un nome de nodo, que ten que ser único na rede. O intercambio de información dentro dunha rede ROS pode ser 1:1, 1:n, n:1 ou n:m.

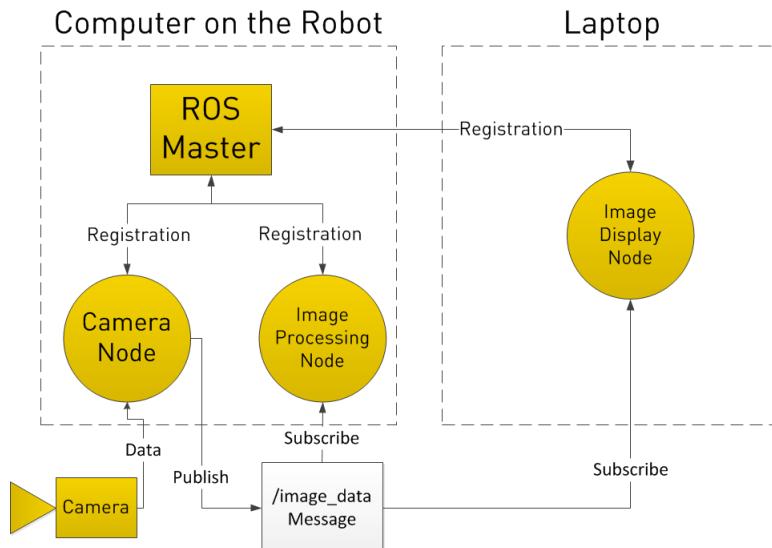


Figura 3.2.3.2 – Exemplo de comunicación entre dispositivos baseada en ROS

A figura 3.2.3.2 mostra un exemplo de configuración dun PC de escritorio que está conectado co robot e serve como o “Master” ROS, o que significa que todos os nodos da rede ROS teñen que rexistrarse nese nodo para poder enviar e recibir datos. Esta configuración inclúe un cliente (portátil) que se rexistra no mestre e subscríbese ao tema *image_data*. Isto significa que unha vez que o nodo de cámara recibe os datos da cámara e os publica nese tema, o

cliente (portátil) recibirá a imaxe e poderá procesala máis. Este procedemento funcionaría de forma análoga para múltiples nodos receptores.

3.3. Python

Python [5] é unha linguaxe de programación amplamente utilizada de alto nivel, de propósito xeral, dinámica, e interpretada cuxa filosofía fai fincapé nunha sintaxe que favoreza un código flexible. A linguaxe proporciona estruturas destinadas a permitir a escritura de programas claros, tanto a pequena como a gran escala.

Python soporta múltiples paradigmas de programación, incluíndo a programación orientada a obxectos, imperativa e funcional ou estilos procesuais. Conta cun sistema de tipo dinámico e xestión de memoria automática e ten unha biblioteca estándar ampla e completa.

En ROS existe rospy, que é unha biblioteca de cliente en Python para ROS. A API (Application Programming Interface) de cliente rospy habilita aos programadores de Python a interconectar rapidamente cos Temas, Servizos, e Parámetros de ROS. Ademais, moitas das ferramentas de ROS, como rostopic e rosservice, son escritas en rospy para aproveitar as capacidades de introspección do tipo.

Na figura 3.3.0.1 móstrase un pequeno exemplo de como se realiza a programación en ROS desde python, onde se pode ver como subscribirse a uns temas e como crear outro para publicar nel.

```
def __init__(self):
    #Initialize ROS publisher, ROS subscriber

    # Subscribe to joint_angles topic
    self.joint_pub = rospy.Publisher('/joint_angles', JointAnglesWithSpeed, queue_size=10)

    self.bridge = CvBridge()

    # Subscribe to joint_angles topic
    self.subs = rospy.Subscriber("/joint_states", JointState, self.joint_states_callback)

    # Subscribe to top camera of the nao_robot
    self.subscriber = rospy.Subscriber("/nao_robot/camera/top/camera/image_raw", Image, self.callback)

    # Publish modified image on a new topic
    self.publisher = rospy.Publisher("/nao_camara_prueba/image", Image, queue_size=10)
```

Figura 3.3.0.1 – Exemplo de programación en ROS a través de Python

3.4. Módulos e librerías adicionais

Á hora de desenvolver as diferentes librerías de interacción empregáronse unha serie de librerías e módulos adicionais que contribuíron na obtención de mellores resultados. Nesta sección explícanse brevemente estas librerías e o seu rol dentro dos subsistemas de interacción desenvolvidos neste traballo.

3.4.1. OpenCV

OpenCV [6] é unha librería de código libre de visión artificial amplamente estendida e cunha gran comunidade detrás. Esta librería orixinalmente está desenvolvida en C++, con todo, existen numerosas interfaces para as linguaxes de programación más comúns. Neste caso úsase a interface a Python.

Open CV é multiplataforma, existindo versións para GNU/Linux, Mac OS X e Windows. Contén máis de 500 funcións que abordan unha gran gama de áreas no proceso de visión, como recoñecemento de obxectos (recoñecemento facial), calibración de cámaras, visión estérea e visión robótica. Esta librería contén utilidades para o traballo con imaxe, tipos de datos específicos e implementacións dos algoritmos más comúns de procesado de imaxes.

Neste traballo é empregada dentro da librería de detección de imaxe, para os módulos de detección de cores e caras, xa que ofrece un conxunto de funcións que permiten realizar distintas tarefas no ámbito da visión artificial.

A maiores está integrada con ROS, o cal, dada a dependencia deste traballo sobre este último, convértea nunha das solucións mellor situadas para abordar este problema. ROS envía as imaxes en mensaxes do tipo *sensor.msg* e co tema *Image*. Para poder utilizar OpenCV de forma conxunta con ROS será necesario utilizar a librería CvBridge, que permite o intercambio de imaxes entre ROS e OpenCV da forma que mostra a figura 3.4.1.1.

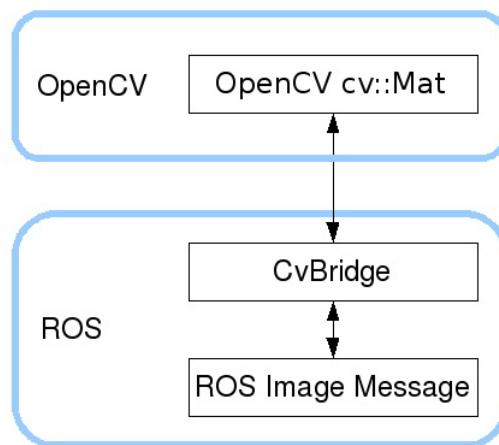


Figura 3.4.1.1 – Comunicación entre OpenCV e ROS por medio da librería CvBridge

3.4.2. APIs e módulos NAOqi

O sistema operativo de Nao, NAOqi, proporciona unha serie de APIs para movemento, son, visión, sensorización, seguidores, percepción da xente, etc. Cada unha destas APIs ten á súa vez unha serie de módulos para traballar determinadas características de cada un destes campos. Por tanto, a continuación, coméntanse os que foron utilizados para este traballo.

- A **API NAOqi Core** inclúe unha lista de módulos básicos que permiten manexar operacións básicas como iniciar e deter un comportamento, administrar a conexión a unha rede, crear módulos propios e máis.

Dentro desta API utilizouse o módulo **ALMemory**, que proporciona unha memoria centralizada que se pode utilizar para almacenar e recuperar valores cun nome. Tamén actúa como un concentrador para a distribución de notificacións de eventos.

- A **API NAOqi Motion** facilita o movemento de Nao. Contén catro grupos principais de métodos para controlar a rixidez articular, a posición da articulación, a marcha, e os efectores do robot no espazo cartesiano.

Dentro desta API empregáronse o módulo **ALMotion**, que proporciona métodos que facilitan o movemento de Nao, e o módulo **ALRobotPosture**, que permite facer que o robot vaia a diferentes posturas predefinidas.

- A **API NAOqi Audio** permite administrar os componentes de software de audio do robot, algúns dos cales inclúen xestión de son, detección e localización de son, e xestión do idioma. Neste caso os módulos da API usados son os seguintes:

- **ALSpeechRecognition** axuda a facer que o robot entenda o que di un humano xa que lle da a capacidade de recoñecer palabras ou frases predefinidas en varios idiomas.
- **ALTextToSpeech** permite que o robot fale.
- **ALSoundDetection** detecta eventos de son.
- **ALSoundLocalization** localiza os sons detectados polo módulo ALSoundDetection, desta forma identifica a dirección de calquera son suficientemente alto oído por Nao.

- A **API NAOqi Trackers** contén tan só o módulo **ALTracker** que permite ao robot rastrexar diferentes obxectivos (bola vermella, cara, punto de referencia, etc.) utilizando diferentes medios (cabeza soamente, corpo enteiro, movemento, etc.).

O obxectivo principal deste módulo é establecer unha ponte entre a detección de obxectivos e o movemento co fin de facer que o robot manteña á vista o obxectivo no centro da cámara.

4 Antecedentes

Como se estableceu nos obxectivos, este TFM céntrase no desenvolvemento de librerías básicas de comportamento para o robot Nao, co obxectivo de que os investigadores do GII poidan utilizaras dentro do proxecto DREAM en tarefas de interacción humano-robot (HRI). Por este motivo, neste capítulo revisase o estado actual da interacción humano-robot (HRI), os diferentes modos de interacción existentes e describense os principais desafíos actuais de investigación neste campo.

4.1. Introdución

Desde finais da década dos 90, a interacción entre humanos e robots cobrou moita importancia, creando un novo campo de investigación na ciencia [7], a interacción humano-robot (HRI polas súas siglas en inglés). O campo da HRI busca entender, modelar e avaliar as diferentes modalidades de interacción entre as persoas e os robots.

Unha primeira clasificación da HRI pode facerse en función da comunicación entre humanos e robots, desta forma pode dividirse en dúas categorías xerais:

- A **interacción remota**, que adoita ser referida como control supervisado ou teleoperación, dependendo de se o robot é autónomo con supervisión dun humano (que intervén en caso de necesidade) ou se o robot é controlado polo humano directamente. Este tipo de interacción pode verse en robots de tipo industrial, como o Baxter, ou en vehículos autónomos, como os vehículos aéreos non tripulados ou UAV.
- A **interacción próxima**, que é aquela na que o robot interactúa directamente co humano, chegando mesmo a haber interacción física. Este tipo de interacción inclúe elementos emotivos e sociais, e pode atoparse en, por exemplo, os robots asistenciais ou educativos. Este tipo de interacción é a que se trata neste traballo.

A HRI é actualmente unha actividade de investigación e deseño moi extensa e diversa. A literatura está a expandirse rapidamente, con centos de publicacións cada ano e coa actividade de moitas sociedades profesionais e reunións ad hoc. Cada ano desde 2006, o IEEE organizou un simposio de especialistas sobre interacción robot-humano [8].

Goodrich e Schultz [7] realizan unha revisión excelente, aínda que lixeiramente anticuada, da literatura existente. Por outra banda, Thomas B. Sheridan tamén realiza a súa revisión sobre

o estado actual e os desafíos da interacción humano-robot [9], na cal indica que se pode facer unha división aproximada da HRI segundo catro áreas de aplicación:

1. **Control a través de supervisión humana de robots no desempeño de tarefas rutineiras.** Estes inclúen o manexo de pezas na fabricación de liñas de montaxe e o acceso e entrega de paquetes, componentes, correo e medicamentos en almacéns, oficinas e hospitais. Tales máquinas poden ser chamadas telerobots, capaces de realizar unha serie limitada de accións de forma automática, baseadas nun programa informático, e capaces de detectar a súa contorna e as súas propias posicións conxuntas e comunicar dita información a un operador humano que actualizará as súas instrucións segundo as necesidades.
2. **Control remoto de vehículos espaciais, aéreos, terrestres e submarinos para tarefas non rutineiras en contornas perigosas ou inaccesibles.** Estas máquinas serán chamadas teleoperadores se realizan tarefas de manipulación e mobilidade nunha contorna física remota seguindo as ordes de control continuo realizado por un humano remoto; mentres que serán chamadas tamén telerobots se un computador é reprogramado intermitentemente por un supervisor humano para executar partes da tarefa en xeral.
3. **Vehículos automatizados nos que un ser humano é un pasaxeiro,** incluíndo automóbiles autónomos, vehículos ferroviarios e avións comerciais.
4. **Interacción social humano-robot,** onde se inclúen dispositivos robot para proporcionar entretenimento, ensino, comodidade e asistencia para nenos e anciáns, persoas autistas e persoas con discapacidade. Nesta categoría atópase o principal enfoque da investigación cos robots humanoides Nao.

A continuación faise un maior fincapé en cada unha das catro áreas, cítanse algúns antecedentes, descríbense investigacións exemplares en curso e discútense desafíos de investigación.

4.2. Control a través de supervisión humana de robots para tarefas industriais rutineiras

Xeralmente, nos procesos industriais, a interacción entre os robots e os operadores adoita ser remota, os sistemas prógramanse para realizar unha tarefa e o humano soamente intervén en caso de necesidade. Con todo, poden darse casos de interacción próxima cos robots. Un destes casos podería ser a aprendizaxe de tarefas mediante demostración, proceso mediante o cal, un operador humano realiza unha tarefa, por exemplo movendo manualmente o brazo dun robot, para que o propio robot aprenda a realizala. Este tipo de interacción permite que os robots aprendan de forma sinxela comportamentos de alto nivel difficilmente programables.

Con todo, á hora de realizar tarefas de forma cooperativa entre robots e humanos, a interacción próxima con robots industriais conleva riscos importantes, como puido verse nun accidente sucedido no 2015 [10] na planta de Volkswagen preto de Kassel, Alemaña, no que un operario foi golpeado por un brazo industrial durante a súa instalación, resultando na morte do técnico. Para evitar esta clase de accidentes, está a buscarse a consciencia da contorna nos manipuladores robóticos, para poder adaptar as súas reaccións ao contexto actual. Este tipo de consciencia non só diminúe os riscos de operación, senón que tamén diminúe os custos espaciais, xa que os robots non requirirán de gaiolas de seguridade.

A produtividade é outro dos factores que se vería afectado positivamente, xa que tarefas imposibles de realizar para un robot e para un humano individualmente, pódense levar a cabo mediante a chamada robótica cooperativa. En Cooperative Tasks between Humans and Robots in Industrial Environments [11] presentan un sistema de robótica cooperativa no que un operador e un robot colaboran de maneira próxima para levar a cabo diferentes tarefas, de maneira que o robot realiza as tarefas repetitivas e perigosas, mentres que o humano leva a cabo as tarefas que requiren de certa precisión ou intelixencia coa que non conta o robot. Neste sistema o operador leva un traxe de posicionamento, que permite ao robot coñecer a súa posición, podendo así adaptar os seus movementos de maneira que o humano non corra riscos.

O robot de liña de montaxe Baxter, mostrado na figura 4.2.0.1, é un produto comercializado por Rethink Robotics [12]. É un robot deseñado para operar de forma segura en estreita proximidade coa xente en contornas de produción sen a necesidade dunha gaiola. Unha innovación interesante no robot Baxter é unha pantalla animada para simular unha cara que lle permite comunicarse co operador humano, mostrar en que está centrado e o seu estado actual, e mesmo pode expresar confusión cando algo non está ben. Ademais, Baxter pode aprender mediante demostración movendo o seu brazo para realizar unha tarefa cun movemento desexado, de forma que pode memorizala e repetila. Isto permítelle aprender a realizar tarefas múltiples e más complicadas sen requerir programación tradicional e en tan só cuestión de minutos, algo que calquera traballador regular podería facer.

Os desafíos actuais de HRI para as tarefas rutineiras esténdense moito máis alá das liñas de montaxe das fábricas e de accións como buscar e entregar pezas e paquetes (exemplo de como o utilizan os almacéns de Amazon), recollida e entrega de correo nos edificios de oficinas, recolección e entrega de medicamentos e subministracións en hospitais, limpeza e tarefas agrícolas automatizadas. As maiores necesidades de investigación neste campo están na planificación, ensino, visualización, control e supervisión das accións automáticas.



Figura 4.2.0.1 – Robot Baxter de Rethink Robotics

4.3. Teleoperación / telerobótica en ambientes perigosos ou inaccesibles

A era da robótica comezou cunha necesidade: como manipular obxectos altamente radioactivos sen expoñer ao operador humano. A finais da década de 1940, Raymond Goertz, no Laboratorio Nacional de Argonne, construíu dispositivos de manipulación remota mestre-escravo mediante os cales se podían captar e mover obxectos nos seis graos de liberdade [13]. Nun primeiro momento, o axuste entre o control mestre do operador humano e o brazo escravo realizouse mediante cintas mecánicas, pero posteriormente este axuste utilizou servomecanismos electromecánicos con retroalimentación de forza.

A día de hoxe realizáronse moitos progresos no control remoto de naves espaciais non tripuladas [14], vehículos robóticos submarinos [15] e vehículos aéreos non tripulados ou UAV [16], aínda que a investigación segue sendo necesaria para mellorar e simplificar as interfaces de visualización e control.

Por outra banda, existen desenvolvimentos prometedores no uso de avatares robóticos para a vixilancia, procura e rescate en traballos policiais, patrullas fronteirizas, loita contra incendios e rescate, e operacións militares [17].

O proxecto DARPA (Defense Advanced Research Projects Agency) [18] promovido polo goberno de Estados Unidos, incentiva a carreira robótica co obxectivo de dar resposta a desastres naturais ou artificiais. As tecnoloxías resultantes da DRC (DARPA Robotics Challenge) transformarán o campo da robótica e catapultarán cara a adiante o desenvolvemento de robots que funcionen e ofrezan a autonomía suficiente para traballar en solitario ou en colabo-

ración con humanos en tarefas comúns nas perigosas condicións das zonas de desastre. Os requisimentos para este tipo de robots, son:

- A mobilidade e destreza para manobrar nos ambientes degradados típicos das zonas de desastre.
- Capacidadade de manipular e utilizar unha variedade diversa de ferramentas deseñadas para os seres humanos.
- Capacidadade para ser manexado polos seres humanos que tiveron pouca ou ningunha formación robótica.
- Autonomía parcial no nivel de tarefa de toma de decisións sobre a base dos comandos do operador e entradas dos sensores.

A fase final do campionato da DRC desafiou aos participantes e os seus robots para completar oito tarefas relevantes para a resposta a desastres, entre elas conducir só, camiñar entre os cascallos, acender interruptores, virar válvulas de roda e subir escaleiras. Ademáis, os patrocinadores de DARPA restrinxiron deliberadamente o ancho de banda da comunicación entre o telerobot e o controlador humano de modo que a teleoperación continua era imposible. Non todos os participantes puideron fazer todas as tarefas, e algúns telerobots caeron e non puideron conseguilo. A figura 4.3.0.1 mostra ao gañador do Instituto Coreano de Ciencia e Tecnoloxía, KAIST, que inxeniosamente tiña pernas con rodas suplementarias unidas.



(a) Robot DRC-HUBO do equipo KAIST

(b) Robot Atlas de Boston Dynamics

Figura 4.3.0.1 – Robots participantes no DARPA Robotics Challenge

Outro dos robots existentes para este tipo de tarefas é o Atlas de Boston Dynamics [19] mostrado na figura 4.3.0.1, un robot humanoide bípedo de alta mobilidade deseñado para negociar terreo ao aire libre. Atlas pode camiñar bípedamente deixando os membros superiores libres para levantar, transportar e manipular o medio ambiente. Nun terreo extremadamente desafiante, Atlas é o suficientemente forte e coordinado como para escalar con mans e pés, para atravesar espazos conxestionados. É por iso que se destina a axudar aos servizos de emerxencia nas operacións de procura e rescate, realizando tarefas tales como pechar válvulas, abrir portas e operar equipos motorizados en contornas onde os seres humanos non poderían sobrevivir. No 2015, o Atlas competiu tamén no Desafío de Robots DARPA para poñer a prueba a capacidade do robot para realizar as diversas tarefas, quedando en segundo lugar por detrás do robot koreano DRC-HUBO por unha marxe de seis minutos.

Os desafíos actuais neste campo de traballo céntranse no continuo desenvolvemento de tarefas como a estimación e a adaptación dos movementos dos robots aos diferentes tipos de terreos aos que se poidan ter que enfrentar, así como a súa estratexia de control, xa que as contornas de operación van a requirir que traballen en terreos accidentados sen quedar atrapados ou volcar no intento. As aplicacións ao aire libre tamén van a facer necesario que os robots posúan certos grados de autonomía para poder operar de maneira efectiva e sen supervisión humana.

4.4. Vehículos autónomos, ferroviarios e aeronaves comerciais

As celebracións anteriores do “Gran Desafío” de DARPA de vehículos autónomos en 2004, 2005 e 2007 demostraron que os vehículos guiados pola intelixencia artificial eran factibles [20]. Como é ben sabido, recentemente Google produciu un automóbil auto-dirixido, demostrado con éxito en autoestradas de California, e en agosto de 2016 a empresa estadounidense nuTonomy, filial do MIT, lanzou o primeiro taxi autónomo do mundo en Singapur [21]. Con todo, máis aló das demostracións de factibilidade, é falaz asumir que un condutor humano se manterá alerta e estará listo para asumir o control nuns poucos segundos se a automatización falla, como puido verse no accidente fatal ocorrido en xuño de 2016 que implicaba a un condutor que usaba o sistema autónomo Tesla Autopilot nun dos coches desta marca [22].

Mentres tanto, moitos outros fabricantes de automóbiles estiveron desenvolvendo tecnoloxía para axudar ao condutor humano, tales como control de cruceiro aumentado por radar, alarmas de saída da estrada e comunicación vehículo-vehículo para previr colisións de intersección. É cada vez más evidente que moitas situacións complexas de tráfico son extremadamente difíciles de entender pola visión e a intelixencia artificias e que moitos accidentes son evitados pola interacción social entre condutores, tales como contacto visual mutuo ou sinais de man [23], [24], [25], [26]. Comprender os aspectos sociais da condución no tráfico, así como o grao en que os automóbiles poden ser automatizados de forma segura, esixe moita más

investigación.

Nesta sección tamén se inclúen avións comerciais porque son telerobots na medida en que os pilotos, sobre todo voando, exercen control de supervisión utilizando o sistema de xestión de voo encargado do control, a orientación e a navegación. Moitas das accións da aviación realizanse por control de lazo pechado, de forma similar a como actúan as habilidades motrices subconscientes de percepción humanas, por exemplo, dirixindo un coche na estrada. A navegación para chegar a un destino aeroportuario é en gran parte unha actividade baseada no coñecemento que se basea na memoria do mapa e os modelos actuais de tempo e tráfico, a miúdo implicando repensar a tarefa para evitar o mal tempo, outras aeronaves ou modificar o horario, de forma similar ao que fan os humanos ao planear un paseo ou a condución a un determinado destino.

Os límites da aviación son actualmente a adopción da navegación GPS, a comunicación dixital co chan e as aeronaves próximas, substituíndo na súa maioría a voz e a visión, e o control aerodinámico e de empuxo adaptativos. Segundo a NASA [27], existe a tecnoloxía necesaria como para retirar o segundo piloto dos avións comerciais, utilizando a monitoraxe terra-base, áinda que a aceptación dos pasaxeiros é unha cuestión diferente e máis desafiante. A investigación neste campo debe incluir simulacións “human-in-the-loop”, nas que un humano sempre forma parte da simulación e, por conseguinte, inflúe no resultado de tal maneira que é difícil, se non imposible, reproducilo exactamente. Tamén deberá avanzarse na avaliación da comunicación dixital en comparación coa comunicación de voz.

4.5. Interacción social humano-robot

Un dos campos nos que a interacción entre humanos e robots cobra moita importancia é no nicho dos robots asistenciais. Estes robots, tamén chamados de servizo, son definidos pola federación internacional de robótica como *Robots que operan de forma total ou semiautónoma para realizar servicios útiles para o benestar de humanos e equipamento, excluíndo as operacións de manufactura* [28]. Nesta definición diferénciase entre dous tipos de robots asistenciais:

- Os **robots persoais**, que son aqueles que se utilizan para labores non comerciais, xeralmente por persoas sen perfil técnico. Por exemplo, robots de asistencia de mobilidade ou aspiradoras automáticas.
- Os **robots profesionais**, que son aqueles utilizados para realizar tarefas de asistencia nunha contorna comercial, xeralmente manexados e supervisados por un persoal especializado. Por exemplo, robots de limpeza automatizados para zonas públicas, robots de mensaxería en oficinas ou hospitais, robots anti-incendios, robots cirúrxicos e de rehabilitación en hospitais ou os robots terapéuticos.

4.5.1. HRI en robots terapéuticos

Nos robots terapéuticos pódense atopar múltiples formas de interacción, por exemplo, o robot Paro [29] é un robot terapéutico con forma de bebe foca, utilizado con éxito en terapias contra a demencia, que busca unha interacción emocional co paciente. Para ese fin conta con cinco tipos de sensores diferentes, táctiles, auditivos, de temperatura, de luz e posturais. Os pacientes realizan unha interacción co robot como a que terían cun animal, e o robot responde acorde aos estímulos que recibe. Isto, en conxunto coa forma física do robot, máis semellante a un animal de peluche que a unha máquina, permite ao paciente desenvolver emocións.

O robot Nao (figura 3.1.0.1) é outro dos robots que se están empregando con éxito en tarefas asistenciais, tanto de maneira terapéutica, como robot de relacións públicas ou para tarefas de educación. O robot conta cunha ampla variedade de sensores e actuadores que lle permiten interactuar de diversas formas co usuario.

Tamén Romeo, mostrado na figura 4.5.1.1, fabricado por Aldebaran Robotics e considerado o irmán maior de Nao, é un robot humanoide duns 140 centímetros de altura deseñado para explorar e profundar na investigación de axudar ás persoas maiores e aqueles que están a perder a súa autonomía. Dentro do proxecto Romeo 2 [30] preténdese ir un paso máis adiante que con Nao, de forma que se realiza investigación en varias áreas técnicas diferentes tales como fiabilidade, percepción multisensorial, interacción cognitiva, interacción física e avaliação. Por tanto, co que Aldebaran e outras institucións francesas tratan de marcar a diferencia neste proxecto é coa súa capacidade para aprender e así, habendo aprendido a aprender, Romeo será capaz de mostrar iniciativa e adaptarse ao hábitos e gustos dos seus propietarios.



Figura 4.5.1.1 – Robot Romeo de Aldebaran Robotics

Por outra banda, conseguir que os pacientes do hospital e especialmente os anciáns confíen nos robots en funcións tales como a asistencia á hora de realización de exercicio ou a entrega de bandexas de alimentos é unha das necesidades da investigación nesta área. As demostracións de Fasola e Matarić [31] e Feil-Seifer e Matarić [32] son exemplares.

4.5.2. HRI en robots cirúrxicos

No ámbito médico, os telerobots cirúrxicos, como o sistema Da Vinci mostrado na figura 4.5.2.1 (aprobado por primeira vez pola Administración de Alimentos e Medicamentos en 2000), permiten agora unha maior precisión dos movementos de man do cirurxián para a ciruxía minimamente invasiva. Neste robot, o cirurxián controla os diferentes brazos do aparello desde unha consola que conta con controis con realimentación háptica, é dicir, o operador non soamente move o brazo, senón que sente o que hai ao final do mesmo, a presión exercida e a resistencia ao movemento, dándolle o tacto necesario para realizar as diferentes tarefas que se realizan nunha operación, como coser ou cortar, de forma natural e cun alto grao de precisión.



Figura 4.5.2.1 – Sistema quirúrxico Da Vinci

Outro dos innovadores robots dedicados á ciruxía é Flex de Medrobotics [33], unha especie de serpe articulada que incorpora unha cámara de alta definición e instrumentos cirúrxicos, capaz de seguir as curvas do organismo. Ofrece así a posibilidade de acceder a localizacións anatómicas que antes eran difíceis ou imposibles de alcanzar de forma minimamente invasiva. Sendo a súa flexibilidade a gran baza ante os robots rígidos, ademais de custar moito menos que Dá Vinci.

A robótica está a transformar o ámbito da ciruxía. Os novos sistemas, cun menor tamaño, revolucionarán os procedementos que requiran accións precisas para acceder a estruturas complexas ou obstruídas dentro do corpo. Un robot máis pequeno permite aos cirurxiáns traballar con varios tipos de ferramentas á vez, conseguindo unha maior proximidade, que permite

aos cirurxiáns realizar incisións más exactas.

Os tamaños redúcense tanto que se comeza a falar de microróbota, e un exemplo disto é Axis desenvolvido por Cambridge Consultants [34], un dos robots máis pequenos coñecidos para uso cirúrxico. Cun corpo externo do tamaño dunha lata de bebidas e instrumentos de só 1,8 milímetros de diámetro, Axis ofrece unha visión do futuro da robótica cirúrxica, onde os microrobots teñen o potencial para revolucionar moitos aspectos da medicina [35]. Estes dispositivos sen tensión, controlados e alimentados de forma inalámbrica farán que os procedementos terapéuticos e de diagnóstico existentes sexan menos invasivos e permitirán novos procedementos nunca antes posibles.

4.5.3. HRI en robots de entretemento

Enténdese por robots de entretemento aqueles cuxa finalidade non é mais que divertir ao usuario. Dentro desta categoría poderíanse incluír aos robots mascota, que xeralmente realizan unha interacción de alto nivel co usuario. Por exemplo, un dos robots mais relevantes no ámbito dos robots mascota é o desenvolvido por Sony, Aibo [36] (figura 4.5.3.1) , cuxa finalidade era comportarse como un can. Neste robot podemos atopar múltiples tipos de interacción: interacción física en forma de movementos perrunos, recoñecemento facial a través de cámaras, a través de caricias utilizando os sensores táctiles, e nas últimas versións do robot mediante unha matriz de leds situada na cara do aparello, que permite poñer diferentes expresións en función do “humor” do robot. Mediante todas estas capacidades motoras e sensoriais, pódese interactuar cunha unidade Aibo de maneira semellante á que se tería cun can real.

Outro exemplo de mascota robótica é o Bandai SmartPet, un robot pensado para utilizar xunto a un smartphone, que é colocado na cabeza do robot e prové múltiples formas de interacción, recoñecemento de xestos mediante a cámara frontal, recoñecemento de son utilizando o micrófono e recoñecemento de xestos táctiles na pantalla.

Por outra banda, unha serie de xoguetes ou figuras humanas que se atopen no mercado incorporan a fala baseada en computador, recoñecemento de voz e software de toma de decisións. Por exemplo, Mattel desenvolveu unha nova boneca Barbie cun extenso vocabulario de recoñecemento do fala e a linguaaxe que está enlazado a través de Internet ao servidor da empresa [37]. A boneca está deseñada para levar a cabo unha extensa conversación con mozas ou mozos en áreas do seu interese. A evidencia suxire que tales dispositivos son eficaces para provocar unha interacción social “normal” con nenos pequenos. Con todo, un pode preguntarse se tales dispositivos inhiben a imaxinación sa (evidenciada polos nenos xogando con bonecas pasivas) en lugar de mellorala [38].



Figura 4.5.3.1 – Robots Aibo de Sony

4.5.4. HRI en robots educativos

Nos últimos anos o uso de robots con fins educativos (aínda que xa eran usados de maneira educativa en ensinanza superior) tomou impulso na educación primaria e secundaria, aparecendo múltiples robots enfocados a este tipo de mercado. A interacción con esta clase de robots pode darse de diferentes formas segundo o público obxectivo, dependendo principalmente do rango de idades do mesmo.

Enfocados á educación infantil pódense atopar robots como o BeeBot ou a súa alternativa libre, o Escornabot. Trátase de robots cuxo labor é o de introducir aos nenos á programación, en forma de pensamento secuencial, e á resolución de problemas. Nesta clase de robots a interacción esta limitada á introdución de comandos na botoeira da parte superior do robot, que serán traducidos a movementos do robot posteriormente. Aínda que esta clase de interacción non é nova, xa que en 1980 a linguaxe LOGO de Papert xa usara “tartarugas” robóticas como medio para que os nenos aprendesen a programación elemental.

Outro robot educativo pensado para ensinar programación aos nenos de diferentes rangos de idade é o Thymio, que permite empregar 3 linguaxes diferentes de programación: Visual Programming Language (VPL) para os nenos a partir de 6 anos; Blockly, unha linguaxe de programación gráfico semellante a Scratch desenvolvida por Google, para nenos a partir de 9 anos; e programación textual mediante o software Aseba Studio para nenos a partir de 12 anos.

Na educación secundaria os robots utilizados xa adquieren unha maior complexidade e adoitan empregarse para unha introdución real á programación, xeralmente utilizando linguaxes gráficas de moi alto nivel como Scratch [39] ou o antes mencionado LOGO [40]. Un exemplo atópase no mBot, que é un kit de robótica para que os nenos se inicien na robótica, programa-

ción e electrónica baseado en Arduino e Scratch, e do cal a Xunta de Galicia comprou 2000 unidades para todos os institutos da comunidade.

Outro dos robots más relevantes neste ámbito é o kit Mindstorms de Lego, mostrado na figura 4.5.4.1, que non só permite programar o robot, senón tamén construílo. O Lego Mindstorms conta con diferentes sensores e actuadores que ofrecen diversas maneiras de interacción co robot como motores, que permiten o movemento do robot de maneira relativamente precisa, sensores de distancia, que permiten medir distancias e actuar en consecuencia aos datos medidos, sensores de luz ambiente, que miden a intensidade da luz da contorna, sensores de cor, que permiten detectar as cores básicas, e unha unidade inercial, que permite medir xiros no plano horizontal.



Figura 4.5.4.1 – Robots creados co kit Mindstorms de Lego

A robótica tamén esta sendo empregada con éxito en actividades de educación especial, por exemplo o robot Nao utilizase para educar a nenos con trastornos de espectro autista. O éxito deste tipo de educación vén dada debido a que a interacción co robot, ao ser programada, resulta previsible para o alumno, creando unha contorna estable e pautada que resulta óptima neste tipo de trastornos. A interacción neste caso dáse en forma de movementos pre-programados e mediante os leds dos ollos do robot, que cambian de cor segundo as emocións que tenta expresar o robot, o cal axuda ao alumno para exercitar un das maiores dificultades dadas polo autismo, a dificultade de establecer relacións empáticas.

4.5.5. HRI con Nao. Uso académico e científico

Desde 2011, máis de 200 institucións académicas de todo o mundo fixeron uso do robot Nao, dende as escolas infantís ata as universidades. Como ferramenta de aprendizaxe, Nao axuda aos estudiantes a aprender sobre a programación, á vez que tamén axuda aos profesor-

res a manter a atención dos seus estudantes mentres ensinan.

O 15 de agosto de 2007, Nao substituíu a Aibo, o can robot de Sony, como o robot utilizado na RoboCup Standard Platform League (SPL), unha competición internacional de robots de fútbol, cuxo obxectivo principal é promover a robótica e a investigación da intelixencia artificial.

Como se comentou na sección anterior, o robot Nao tamén se utiliza para educar a nenos con trastornos de espectro autista. Desta forma, en 2012, unha serie de robots de Nao foron empregados para ensinar a nenos autistas nunha escola do Reino Unido e, segundo as probas que emerxen dese ensaio, parece que os nenos poden aprender mellor dos robots que dos profesores humanos, xa que algúns deles atoparon os robots de expresión infantil más expresivos que os propios seres humanos [41], [42]. Nun contexto máis amplio, os robots Nao foron utilizados por numerosas escolas británicas para introducir aos nenos aos robots e a industria da robótica [43].

Por outra parte, tamén os Nao son utilizados con fines terapéuticos. Un exemplo disto atópase nun estudo realizado nun centro para maiores en España [44], no cal se introduciu o robot Nao para asistir nas clases regulares de fisioterapia. O rol de Nao era modelar os movementos do fisioterapeuta para os pacientes. Os resultados do experimento deixaron ver que os pacientes adaptaban os seus movementos aos de Nao, á vez que ao fixarse no robot melloraban a calidade técnica de execución dos propios exercicios. Un experimento similar foi tamén realizado en 2015 en Molineaux, Francia [45]. A figura 4.5.5.1 mostra a realización dunha destas probas.



Figura 4.5.5.1 – Uso de Nao como robot terapéutico

En 2015, Mitsubishi UFJ Financial Group comezou a probar robots Nao para o uso de servizo ao cliente nas súas sucursais bancarias xaponesas [46]. En xullo de 2015, os robots Nao

foron usados para demostrar unha forma básica de autoconciencia nun experimento filosófico no Rensselaer Polytechnic Institute de Nova York [47]. En setembro de 2015, o Instituto Francés de Saúde e Investigación Médica utilizou robots Nao para probar un sistema de “memoria autobiográfica” robótica deseñada para axudar a formar tripulacións da Estación Espacial Internacional e asistir a pacientes anciáns [48].

Na actualidade Nao está dispoñible como un robot de investigación para escolas, institutos e universidades para ensinar programación e realizar investigacións na área da interacción humano-robot. É por iso que xa se está utilizando para fins específicos de investigación e educación en numerosas institucións académicas de todo o mundo, e tal é o seu impacto que, a data de 2015, máis de 5000 robots están en uso en preto de 70 países [49].

4.6. Conclusión

Como se puido apreciar nas seccións anteriores deste capítulo, a HRI é un campo amplo e de rápida evolución e, polo tanto, crucial na evolución da robótica.

Os robots especializados en teleoperación humana demostraron ser exitosos en ambientes perigosos e aplicacións médicas, do mesmo xeito que os telerobots especializados na realización de tarefas industriais repetitivas. A investigación nas áreas dos automóbiles autónomos, a colaboración íntima cos seres humanos en tarefas de manipulación, o control humano de robots humanoides para ambientes perigosos, e a interacción social con robots está nas etapas iniciais. Mientras que a eficacia dos robots humanoides de propósito xeral aínda non foi probada.

Mientras que a raza humana está a cambiar moi lentamente, os computadores e os robots están a evolucionar a un ritmo moi rápido. Por tanto, é probable que as conclusións específicas sacadas sobre HRI se volvan inválidas en pouco tempo. É por isto que a motivación da comunidade de HRI parece estar más enfocada en construír e demostrar o que funciona e provocar novas ideas que en prover conclusións científicas detalladas e validadas.

No que respecta ao robot Nao, que será utilizado no presente TFM, queda patente o seu gran potencial como ferramenta de interacción social dado o seu aspecto humano, que lle proporciona unha gran empatía cos usuarios. Esta é a principal razón do seu uso no proxecto DREAM de cara a recabar datos de interacción.

5 Requisitos de deseño

A continuación describense os requisitos para o desenvolvemento dos comportamentos de máis alto nivel (seguir un obxecto de cor, coller un obxecto, atender a sons, etc) que serán realizados neste traballo:

- En primeiro lugar, tal e como está establecido dentro do proxecto DREAM, todos os desenvolvimentos que se realicen deberán atoparse baixo os estándares de **ROS** (Robot Operating System), polo que será necesario implementar os devanditos comportamentos básicos neste novo sistema.
- Deberá empregarse a linguaxe de programación **Python**, xa que é a linguaxe utilizada orixinalmente nas ferramentas e aplicacións proporcionadas pola empresa Aldebaran para traballar cos robots humanoides Nao. Ademais Python é a linguaxe más utilizada por outros desenvolvedores dos mesmos robots, o que facilitaría a incorporación de novas funcionalidades futuras desenvolvidas por outros se así se desexase.
- Por outra parte, tamén haberá que facer uso da **API de NAOqi** coa que conta Nao, xa que a parte de empregar as librerías desenvolvidas en Python, tamén son habituais os códigos que combinan esta API xunto con paquetes de ROS. Demostrarase desta forma que ambos son compatibles.
- Por último, todas as librerías de comportamentos básicos desenvolvidas deberán realizarse en forma de scripts para a súa rápida execución. Tamén deberán estar correctamente documentadas e contar con exemplos de aplicación para facilitar a súa comprensión.

6 Desenvolvemento das solucións

Neste capítulo desenvolveranse os detalles da implementación de cada unha das librerías creadas para favorecer a interacción humano-robot a través de Nao, así como a metodoloxía de traballo seguida para chegar ás solucións.

Definirse o sistema global que forma o traballo, tratando primeiramente os seus principais componentes e rematando pola definición do sistema de comunicación. E continuarase co desenvolvemento de cada unha das librerías de comportamento básico creadas para a interacción con humanos: a librería de detección de imaxe, a librería de son e a correspondente aos movementos básicos do robot.

6.1. Arquitectura global

Esta sección describe como todos os componentes do sistema traballan xuntos e resume os fluxos de datos entre eles. A arquitectura final da implementación reúne todas as funcionalidades e algoritmos, e está deseñada para dar forma e apoiar a colaboración entre os tres actores involucrados no proceso de funcionamiento das diferentes librerías: o usuario, o robot e o sistema informático, tal e como se mostra na figura 6.1.0.1.

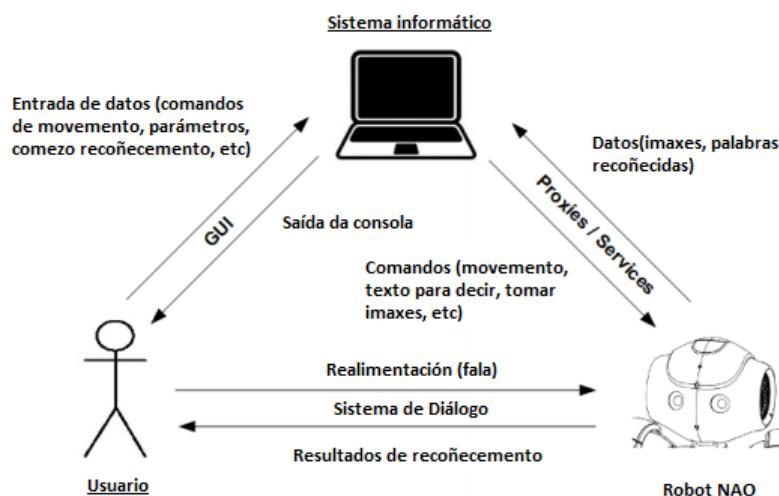


Figura 6.1.0.1 – Compoñentes involucrados no funcionamiento das diferentes librerías

- O sistema informático serve como interface de comandos entre o usuario e o robot, e

realiza os cálculos de recoñecemento de obxectos reais, sons e movementos.

- O robot toma imaxes coas súas cámaras, sons cos seus micrófonos, informa dos resultados usando o sintetizador de voz, proporciona información acerca das posicións das súas diferentes articulacións, e demais datos tomados a través dos seus sensores.
- O usuario opera e controla todo o sistema e pode proporcionar retroalimentación utilizando o recoñecemento de voz do robot.

Cabe sinalar que, aínda que teoricamente é posible realizar cálculos de recoñecemento de obxectos, sons, etc, no propio robot Nao, as súas capacidades computacionais son insuficientes para obter resultados o suficientemente rápidos, motivo polo cal se realizan todos nun computador externo.

Por outra banda, como se comentou inicialmente, a lingua de programación empregada é Python, xa que permite:

- crear comportamentos propios accedendo directamente ás APIs de NAOqi, que proporcionan acceso de baixo nivel aos sensores e actuadores do robot.
- acceder aos distintos temas e servizos que se atopan dispoñibles en ROS, que en esencia son as partes necesarias das APIs de NAOqi proporcionadas por Aldebaran, pero coas vantaxes que proporciona o Sistema Operativo de Robots.

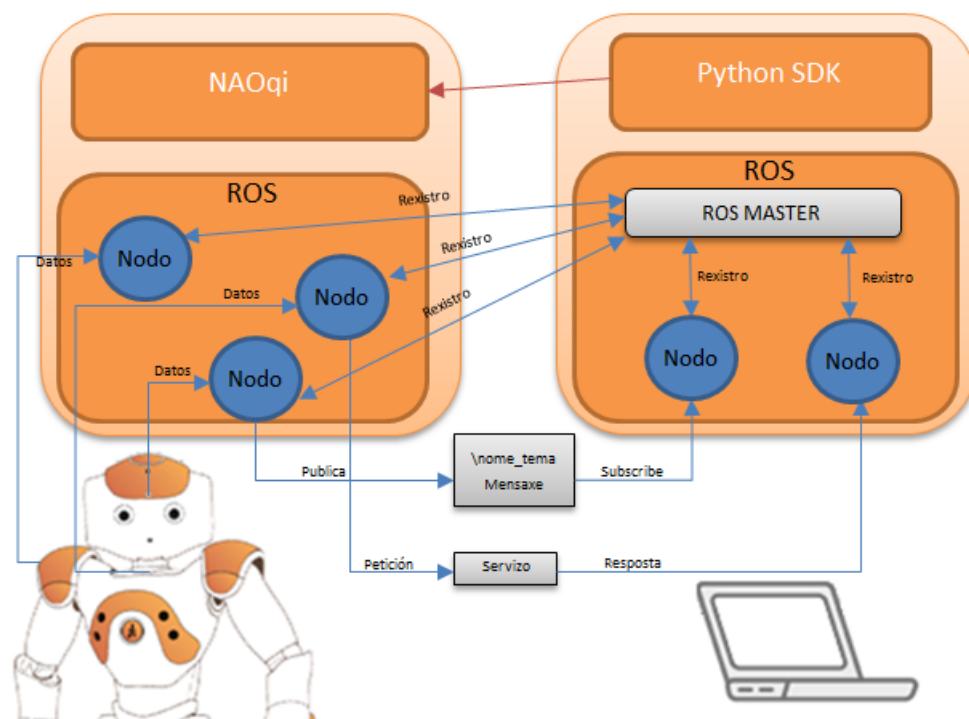


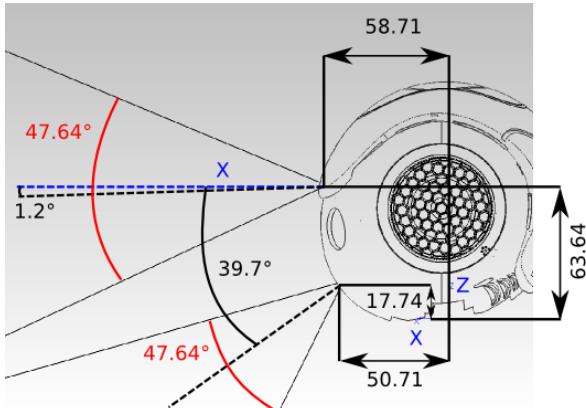
Figura 6.1.0.2 – Esquema de comunicación Nao-Python-ROS

Por tanto, faise necesario definir tamén o sistema de comunicación Nao-Python-ROS, que ao fin e ao cabo será o que permita desenvolver as diferentes librerías e tratar de aproveitar as vantaxes que proporciona cada unha das partes. Devandito sistema móstrase na figura 6.1.0.2 e permite visualizar de forma clara como se produce a comunicación entre o robot e o computador externo, xa sexa a través dos módulos incluídos en NAOqi, ou a través dos temas e servizos de ROS; e todo xestionado a través da linguaxe de programación Python.

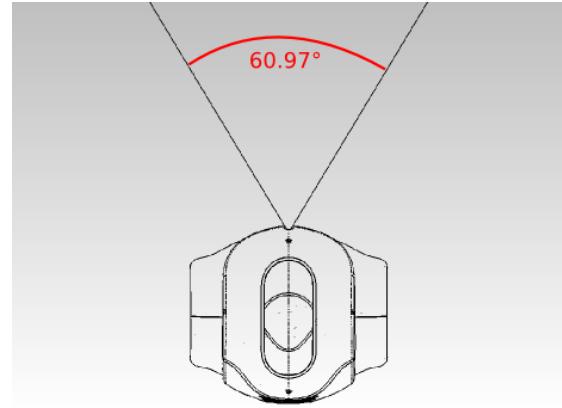
6.2. Librería de detección de imaxe

Unha das partes principais da interacción humano-robot é a percepción da contorna por parte do robot. A utilización de cámaras co consecuente tratamento da imaxe é a parte principal desta percepción, de forma que o robot poida responder en función do que está a ver. Isto permitiralle asemellar o seu comportamento máis a como o fan os humanos, de forma que poida recoñecer unha cara, unha cor, unha forma ou unha etiqueta e centrar a súa atención nela.

Para a detección de imaxe farase uso da cámara superior do robot Nao, por considerarse que permite unha visión más ampla da contorna que o rodea. A cámara atópase na parte superior da súa cabeza e sen que o robot vire a cabeza ofrece un amplio campo de visionado, de aproximadamente 61° , da forma mostrada na figura 6.2.0.1.



(a) Posición das cámaras superior e inferior



(b) Campo de visión

Figura 6.2.0.1 – Posicionamento e campo de visión das cámaras situadas na cabeza de Nao

Os algoritmos para o tratamiento da imaxes e a posterior detección das características desexadas, serán diferentes para o caso da detección de cor e para o caso de detección de caras, aínda que ambos se implementarán coa axuda da librería de visión artificial OpenCV.

Con un ou outro algoritmo o que se tratará de facer é detectar o punto central do obxecto de cor ou a cara desexados e, a partir das coordenadas dese punto e coa axuda da librería de movemento, tratar de mover a cabeza ou o corpo de forma que o punto estea sempre situado no centro da imaxe vista a través da cámara superior do robot.

6.2.1. Detección de obxectos de cor

Existen infinidade de traballos nos que se poden detectar distintos obxectos ben pola forma do obxecto ou ben pola cor que teña. Neste caso, como o obxectivo será recoñecer unha pelota de cor, o que se fará será unha combinación de ambas as dúas técnicas, recoñecendo primeiro a cor desexada na imaxe e buscando despois que se trate dunha forma circular. Polo tanto, esta funcionalidade da librería de detección de imaxe pode dividirse en dúas etapas:

1. Detección de cor.
2. Detección de formas circulares.

6.2.1.1. Detección de cor

O primeiro que hai que considerar á hora realizar o recoñecemento de cor é determinar o modelo de cor que se vai a utilizar. Neste caso, como se explicará máis adiante, o elixido será o modelo HSV, xa que é ou que emprega a librería de recoñecemento de cores contida en OpenCV.

O obxectivo principal desta primeira etapa será determinar a cor do obxectivo, neste caso unha pelota. Desta forma, e co posterior aseguramento da súa forma circular, farase que a detección sexa moito más robusta que se só se fixese algunha das cousas por separado, evitándose á vez gran cantidade de erros.

A detección de cor no rango HSV vai a realizarse a partir da imaxe obtida a través da cámara superior, dándose por feito que o obxecto será mostrado previamente ao robot para así poder extraer as súas características. Noutro caso habería que introducir directamente os valores HSV no código do programa, podendo producirse erros se se producen cambios na iluminación ambiente ou posibles fallos na introdución dos números.

As accións a realizar pódense resumir da seguinte forma:

1. Obtención da imaxe a partir da cámara superior de Nao.
2. Delimitación do contorno e do centro da pelota para delimitar en que zona coller os píxeles.
3. Calcular a media HSV dos píxeles elixidos.
4. Determinación do rango de detección a partir do valor HSV calculado.

Como ben se dixo, en primeiro lugar obtense unha imaxe da cámara para realizar a detección de cor da pelota.

A continuación, para delimitar a zona circular a partir da cal extraer os píxeles para determinar a cor do obxecto, faise uso da función **HoughCircles** de OpenCV, que permite atopar círculos nunha imaxe en escala de grises utilizando unha modificación da transformada de Hough. Normalmente esta función detecta ben os centros dos círculos. Con todo, pode fallar á hora de atopar os radios correctos. Para iso, é posible ‘axudala’ especificando os radios mínimos e máximos de alcance que, a priori, deben ser coñecidos porque se sabe cal é o obxecto que se desexa localizar.

Tamén haberá que ter coidado cos parámetros que establecen a mínima distancia entre os círculos a detectar, xa que se este valor non é correcto, poderase acabar detectando moitos círculos falsos ou deixando de detectar algúns que si son correctos. Por tanto, é necesario un axuste fino antes de utilizar esta función na librería de comportamento básico.

Un exemplo do uso desta función é o mostrado na figura 6.2.1.1 onde a posición da pelota, sobre os brazos estirados do robot, asegura que non existan máis formas circulares na imaxe e que a contorna da bola se vexa perfectamente. Desta forma a detección será más precisa e poderanse extraer dela os píxeles necesarios para determinar a súa cor.

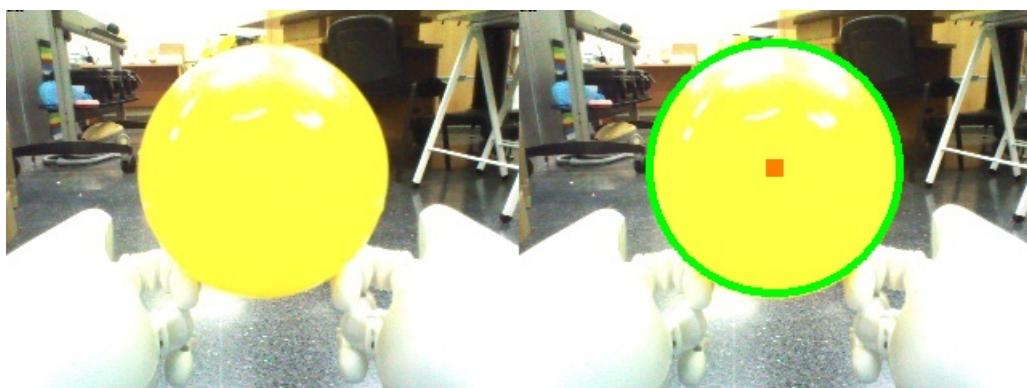


Figura 6.2.1.1 – Uso da función ‘HoughCircles’ para o recoñecemento da pelota

A maiores, ao recoñecer o círculo debuxarase sobre a imaxe o seu centro e a súa contorna para asegurar desta forma que o obxecto recoñecido é a pelota, e para que o usuario poida comprobar que realmente o recoñecemento da cor non é froito da casualidade.

O seguinte paso será converter a imaxe ao espazo de cor HSV, para o que, sabendo as coordenadas do centro da pelota na imaxe, se calcula a media dos valores HSV dentro dun rectángulo de 45 píxeles de lado, próximo ao centro da pelota detectada. O rectángulo laranxa mostrado na figura 6.2.1.2 corresponderíase cos píxeles tomados para calcular a media do rango de cor, dado que ocupan a maior porcentaxe da área vista da pelota. Realízase a aproximación desta forma porque na parte superior da pelota poden existir reflexos debidos á luz existente na sala que falseen a medida e produzan un rango de detección de cor falsa.

Por último, unha vez coñecido o valor medio dos valores HSV dentro da pelota establecense



Figura 6.2.1.2 – Rectángulo representativo dos píxeles utilizados para a estimación

uns límites superior e inferior para facer más robusta a detección, é dicir, que se a bóla se atopa nunha sombra ou nunha situación de maior claridade ca cando foi calibrada a súa cor, esta sea recoñecida de igual forma. Estes límites calcúlanse da seguinte forma:

$$\begin{aligned} \text{Límite superior} &= [H + 10 \quad 255 \quad 255] \\ \text{Límite inferior} &= \begin{cases} [H - 10 \quad 40 \quad 40] & \text{se } S < 100 \\ [H - 10 \quad 100 \quad 100] & \text{se } S \geq 100 \end{cases} \end{aligned} \quad (6.2.1.1)$$

sendo H e S os valores medios de matiz e saturación dos valores HSV antes calculados.

Haberá que ter en conta para este cálculo que, debido ao algoritmo de autoexposición co que conta a cámara, cando se mostren imaxes de cores próximos ao violeta, esta cambiará a tonalidade da imaxe. É por iso, que nos casos en que o valor de Saturación é menor que 100, cambian os valores do límite inferior, da forma que indica a ecuación 6.2.1.1. Hai que engadir que estas formas foron axustadas despois da realización de moitas probas.

Na táboa 6.2.1.1 atópanse recollidos os valores medios HSV calculados para cada unha das cores de pelotas existentes no laboratorio onde se realizaron as probas. Como se pode apreciar, estes rangos atópanse moi preto uns dos outros no caso de cores como o vermello ou o laranxa. Desta forma vese que se existen obxectos de cores similares na mesma sala ou no lugar onde o robot realice o seu traballo, é probable que se poidan confundir. De aí a necesidade de facer más robusta a detección ao engadir tamén a parte de aseguramento da forma circular do obxecto, e non só a súa cor.

Hai que comentar que, nun primeiro momento, a detección de cor facíase de forma manual e indicábanselle ao robot directamente no código os rangos de cores a seguir. Sen embargo, finalmente, e para previr posibles error por mor de que as condicións de iluminación poidan cambiar respecto ao momento no que se fixo a calibración manual, optouse por que o robot realice el mesmo dita calibración ao ver a pelota no momento inicial. Comportamento co cal

Cor	Límite HSV Superior	Límite HSV Inferior
Amarelo	[40 255 255]	[20 100 100]
Azul	[110 255 255]	[90 100 100]
Laranxa	[30 255 255]	[10 100 100]
Verde	[95 255 255]	[75 100 100]
Vermello	[20 255 255]	[0 100 100]
Violeta	[150 255 255]	[130 40 40]

Táboa 6.2.1.1 – Límites HSV medios para diferentes cores

tamén se asemella ao que faría calquera ser humano, de forma que se fai un avance en canto a obter un comportamento más real para a interacción humano-robot.

6.2.1.2. Espazo de cor HSV

A continuación realizase unha breve explicación acerca do modelo de cor HSV, xa que é o que se emprega na librería de recoñecemento de cores contida en OpenCV.

O modelo HSV (do inglés Hue, Saturation, Value - Matiz, Saturación, Valor) define un modelo de cor en termos dos seus compoñentes. O espazo de cor HSV ten tres compoñentes: matiz, saturación e valor. O “valor” substitúese ás veces polo “brillo” e entón coñécese como HSB. Outra forma de chamar a este modelo é como o modelo de cor de cono hexagonal.

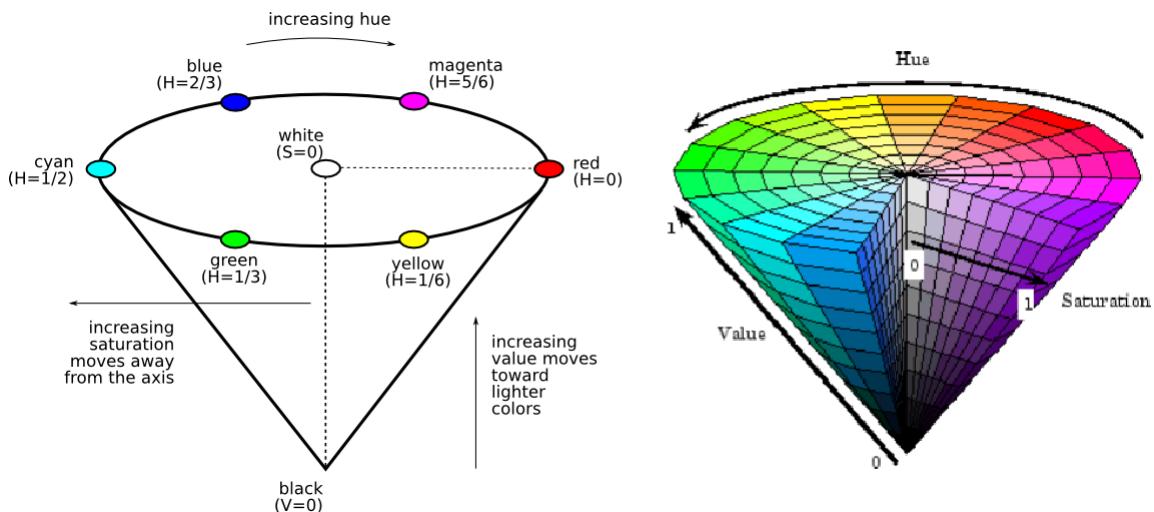


Figura 6.2.1.3 – Diagrama do espazo de cor HSV

- O **matiz** (H) dunha cor refírese á cor pura á que se asemella. Por exemplo, todos os tons de vermello teñen o mesmo matiz. As tonalidades son descritas por un número que especifica a posición da cor pura correspondente na roda de cor, como unha fracción entre 0 e 1. O valor 0 refírese ao vermello, 1/6 é amarelo, 1/3 é verde e así sucesivamente ao redor da roda.
- A **saturación** (S) dunha cor describe a “brancura” da cor. Por tanto cando o valor de saturación é 0 a cor é branco, mentres que se o valor é 1 tratarase dunha cor primaria.

Por exemplo, un vermello puro está totalmente saturado (saturación de 1). Á súa vez as diferentes tonalidades de cor vermella terán unha saturación inferior a 1. Canta maior saturación exista nunha cor más viva será a mesmo, mentres que a menos saturación más grisácea será.

- O **valor** (V) dunha cor, tamén chamado brillo, describe a escuridade da cor. Un valor de 0 levará a unha cor negra, mentres que ao crecer o brillo afastarase do negro e aparecerán as diferentes cores.

O diagrama da figura 6.2.1.3 pode axudar a visualizar mellor o significado dos parámetros H, S e V.

En visión por computador deséxase separar os compoñentes da cor da intensidade por varias razóns, como poden ser a robustez aos cambios de iluminación ou a eliminación de sombras. Por tanto, sendo estes os obxectivos, pódese obter mellor información dun espazo de cores HSV que dun RGB, onde os seus compoñentes están correlacionados coa cantidade de luz que golpea ao obxecto, e por tanto entre sí, o que dificulta a discriminación de obxectos.

6.2.1.3. Detección de formas circulares

A procura de formas circulares baséase na detección dos píxeles dunha cor determinada (que xa se coñece da etapa anterior) na imaxe dada pola cámara.

Estes píxeles filtranse dependendo da súa distancia ao valor da cor desexada no espazo de cor HSV, utilizando un límite previamente calculado para permitir a detección mesmo con condicións de luz cambiantes. Entón, de todo o conxunto detectado de píxeles da devandita cor, só se manteñen os que definen unha forma circular. De forma similar, pódese definir a amplitude do obxecto e o seu tamaño mínimo.

Para detectar a pelota, simplemente se tratará de buscar os contornos existentes na imaxe. Estes contornos pódense explicar simplemente como unha curva que une todos os puntos continuos (ao longo do límite), que teñen a mesma cor ou intensidade, polo que se converten nunha ferramenta útil para a análise de formas e a detección e recoñecemento de obxectos.

En OpenCV existe unha función chamada **FindContours** que permite atopar os contornos existentes nunha imaxe en branco e negro. Por tanto será necesaria a aplicación previa dunha máscara cos límites de cor a detectar, de forma que o obxecto a buscar sexa branco e o fondo sexa negro.

De forma resumida, a forma de operar é a seguinte:

1. Convértese a imaxe da cámara ao espazo de cor HSV.

2. Constrúese unha máscara para a cor desexada e, a continuación, realízanse unha serie de dilatacóns e erosóns para eliminar as pequenas imperfeccións que quedasen na máscara.
3. Búscanse os contornos na máscara e obtense o centro (x, y) da pelota.

Na figura 6.2.1.4 pódese ver o resultado de aplicar a máscara de cor sobre a imaxe orixinal e a posterior detección do contorno circular sobre a propia máscara.



Figura 6.2.1.4 – Exemplo de aplicación de máscara e posterior procura do contorno

Para poder tratar a imaxe o primeiro paso será converter o frame obtido da cámara superior do robot, a través do correspondente tema de ROS, de formato de mensaxe ‘Image’ de ROS a un formato de mensaxe en OpenCV.

A continuación convértese a imaxe ao espazo de cor HSV e constrúese unha máscara para a cor desexada, que, tal e como se dixo, foi previamente detectada ou indicada dalgunha maneira polo usuario ao robot, e realízanse unha serie de erosóns e dilatacóns na imaxe para eliminar as pequenas imperfeccións que poidan quedar na máscara.

Unha vez realizado este paso búscanse os contornos na máscara creada e, tras asegurar que polo menos un contorno foi atopado, búscase o maior dos existentes e utilizase para calcular un círculo que o encerre e o seu centroide. Desta forma preténdese que no caso de que existan varias pelotas da mesma cor se siga a de maior tamaño ou, na súa falta, a que se atope más proxima ao robot, tal e como se ve nas figuras 6.2.1.5 e 6.2.1.6.

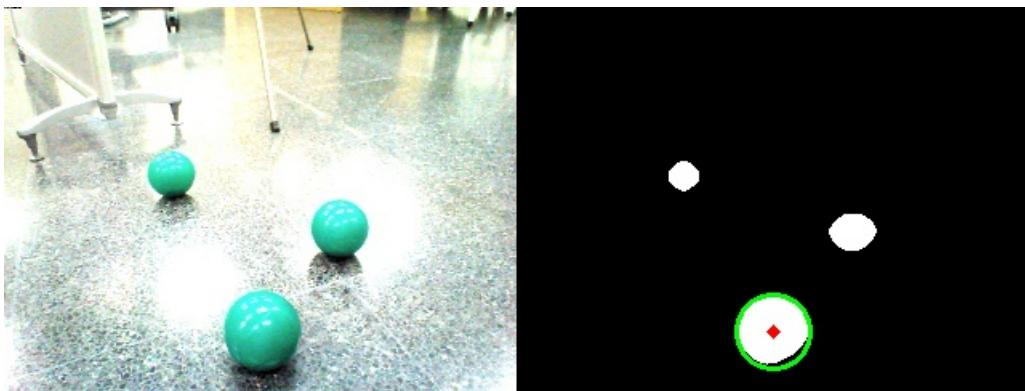


Figura 6.2.1.5 – Exemplo de detección cando existen varias bolas da mesma cor

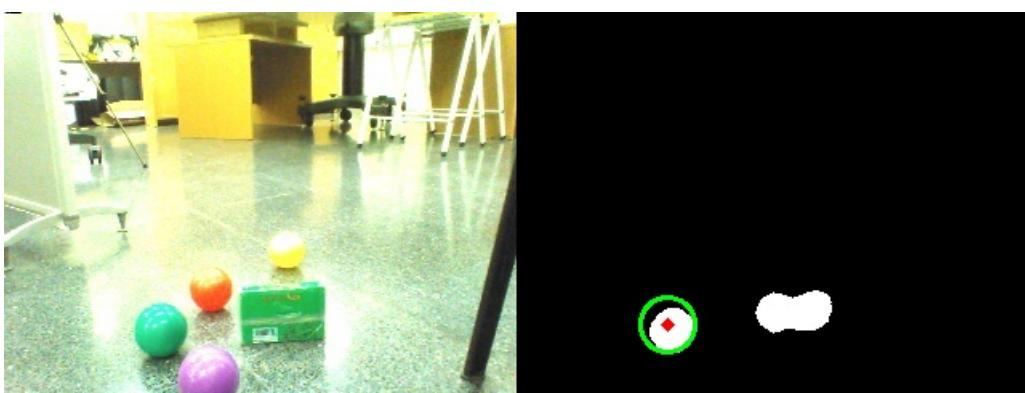


Figura 6.2.1.6 – Exemplo de detección cando existen varias bolas da diferentes cores

Tamén servirá para poder previr fallos na identificación, como no caso da figura 6.2.1.7 onde se mostra que, a maiores da pelota, existe algúun outro obxecto da cor indicada e, polo tanto, ao detectar o contorno circular poderase distinguir o obxecto que realmente é o obxectivo doutros que non o son.



Figura 6.2.1.7 – Exemplo de detección con obxectos da mesma cor na imaxe

O seguinte paso será comprobar que o radio do círculo é maior dun determinado valor, para asegurar desta forma que a forma circular atopada realmente pode ser unha das pelotas utilizadas (das cales o radio é coñecido).

Seguidamente debuxarase o círculo e o centroide da pelota sobre a imaxe orixinal. Esta imaxe modificada será publicada noutro tema de ROS para que sirva como realimentación ao usuario e que así poida comprobar que o que realmente se está detectando é a pelota da cor desexada, da forma mostrada na figura 6.2.1.8. Previa á publicación da imaxe no tema de ROS será necesario volver facer unha transformación de formato de mensaxe OpenCV, que é co que se traballou, a formato de mensaxe ‘Image’ de ROS.



Figura 6.2.1.8 – Exemplo da imaxe publicada para realimentación ao usuario

Por tanto, unha vez finalizada a detección xa se dispón das coordenadas do centroide da pelota, que é o obxectivo principal de todo este proceso.

6.2.2. Detección de caras

No caso de recoñecemento de caras o algoritmo utilizado xa traballa dunha forma distinta a como o facía cando se estudaba a detección de obxectos de cor, neste caso ou seu funcionamento baseárase non recoñecemento a través de fervenzas de ‘Haar’.

A detección de obxectos utilizando os clasificadores de fervenzas de ‘Haar’ é un método de detección efectivo proposto por Paul Viola e Michael Jones [50]. Esta funcionalidade realiza tamén coa axuda de OpenCV, que traballa utilizando ditas ‘fervenzas’, as cales se basean en detectar caras en múltiples etapas.

Para cada subrexión dunha imaxe faise unha proba aproximada e rápida, e se esta proba se pasa, faise unha proba un pouco máis detallada, e así sucesivamente. O algoritmo pode ter 30-50 destas etapas ou fervenzas, e só detectará unha cara se todas as etapas se pasan. Na figura 6.2.2.1 móstrase de forma conceptual a forma de traballo das diferentes fervenzas.

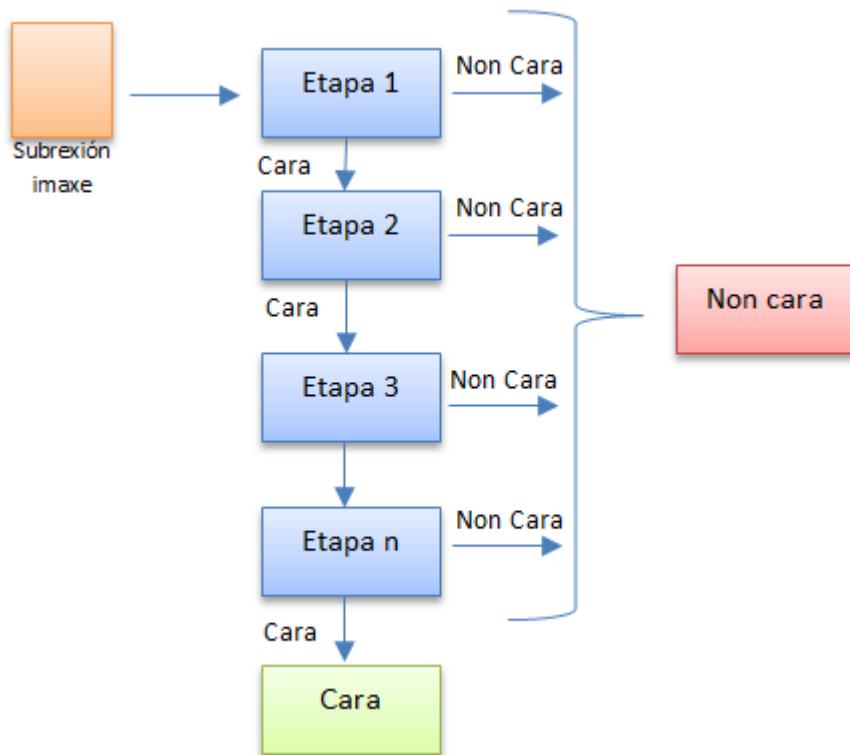


Figura 6.2.2.1 – Funcionamento dun detector baseado en multiples etapas ou fervenzas

As fervenzas son só unha chea de arquivos de tipo XML que conteñen datos OpenCV utilizados para detectar obxectos, por tanto basta con inicializar o código coa fervenza que se desexa e logo esta xa fai o traballo. Neste caso, como se comentou xa, a fervenza utilizada é de tipo ‘Haar’ e serve para recoñecer caras que están de fronte á imaxe, pero existen outros moitos tipos delas.

Hai que comentar tamén que estes algoritmos contan cunha serie de parámetros que haberá que axustar en función do ambiente no que se pretendan empregar, isto é, non será o mesmo recoñecer unha cara nun ambiente sombrío que nun con moita luz, ou se a cara se atopa máis preto ou máis lonxe da cámara. Os principais parámetros son os seguintes:

- O factor de escala, utilizado para compensar o feito de que algunas caras poden estar máis preto da cámara que outras.
- O tamaño da xanela móvil utiliza para definir cada unha das subrexións da imaxe.
- O número de obxectos que se detectan preto da xanela actual antes de declarar a cara como ‘atopada’. Este parámetro afectará á calidade das caras detectadas: un valor máis alto resulta en menos deteccións pero con maior calidade.

Por outra banda, o algoritmo devolve unha lista de rectángulos onde cre que atopou unha cara, os cales están definidos por catro valores que son: a localización ‘x’ e ‘y’ do rectángulo, e o ancho e altura do mesmo ‘w’ e ‘h’. Un exemplo do seu funcionamento, unha vez implemen-

tado no robot, pode verse na figura 6.2.2.2.

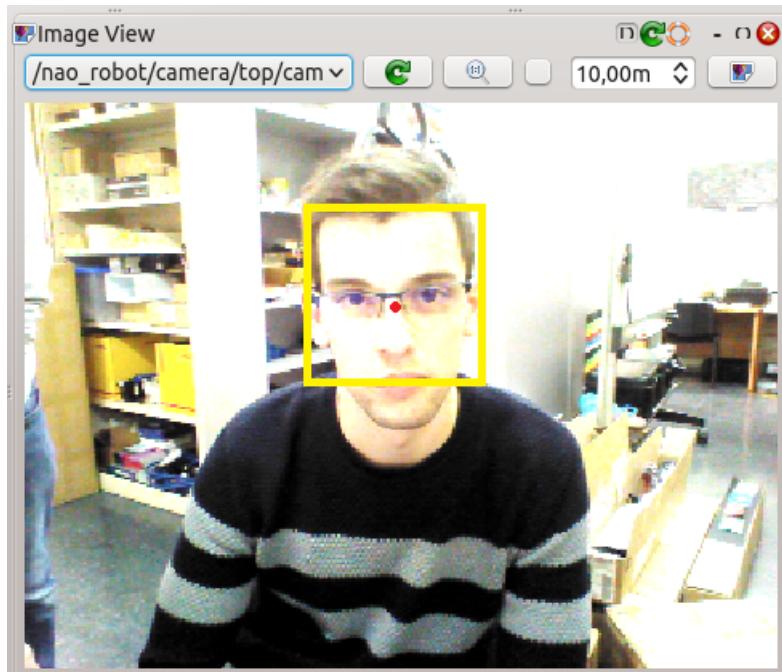


Figura 6.2.2.2 – Exemplo de detección de caras

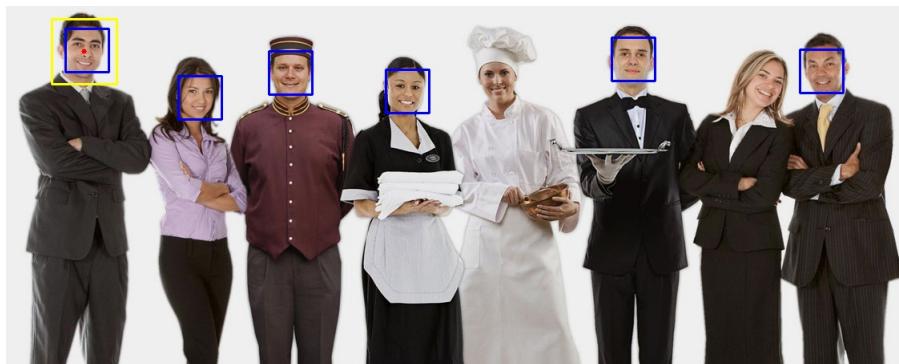
Á hora da implementación da librería de detección facial, procederase ao axuste dos principais parámetros do algoritmo. Destes parámetros, axustaranse o factor de escala e número de obxectos que se detectan preto da xanela actual, xa que, como se dixo con anterioridade, inciden directamente na calidade da detección.

Nun primeiro momento tende a pensarse que se indica a máxima calidade para a detección de caras, tan só haberá que axustar o factor de escala para obter unha solución óptima, pero non é así. Xa que se o recoñecedor é moi preciso tardará moito tempo en executarse, e dado que neste caso o que se pretende é poder empregalo en tempo real, iso non sería unha posibilidade. Inda que, por outra banda, se o detector é pouco preciso executarase en menos tempo, pero pode non chegar a recoñecer moitas caras reais.

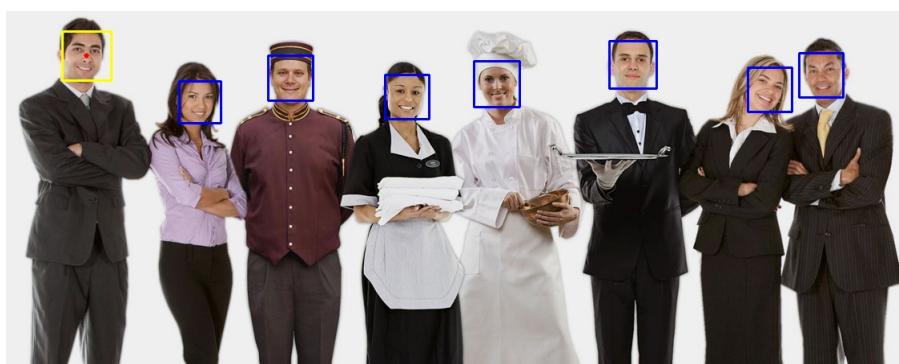
Será necesario, polo tanto, chegar a un punto intermedio de consenso, onde o parámetro de calidade permita á función traballar o suficientemente rápido como para poder executala nunha aplicación que traballa en tempo real, de forma que permita o seguimento dunha cara, e que teña a suficiente precisión como para detectar as caras más básicas ou próximas á cámara do robot, xa que tampouco se trata de as identifique todas, porque só se vai a seguir a cara principal, é dicir, a de maior tamaño, xa que, a priori, será a más próxima ao robot e a que pretenda interactuar co mesmo.

Unha vez axustado este parámetro, bastará con axustar o factor de escala para obter o mellor resultado posible.

A continuación inclúense un exemplos acerca de como afecta o factor de escala á detección, unha vez a calidade da mesma xa foi axustada. Así, na parte superior da figura 6.2.2.3 pode verse como cun factor de escala maior hai caras que quedan sen detectar, mentres que se este diminúe o seu valor ata un punto concreto, no cal segue sendo rápida a execución do algoritmo, é posible detectalas todas, tal e como se mostra na parte inferior da mesma figura.



(a) Factor de escala maior - Menor tempo de execución pero peor resultado



(b) Factor de escala menor - Resultado óptimo dentro da marxe de tempo

Figura 6.2.2.3 – Influencia do factor de escala no algoritmo de recoñecemento facial

Hai que comentar que á hora da utilización deste algoritmo dentro da correspondente librería, terase en conta tan só a cara de maior tamaño, que é a que nos exemplos da figura 6.2.2.3 se mostra encadrada dentro dun rectángulo de diferente cor e cun círculo de cor vermello indicando o centro do mesmo. Isto faise para indicar que ese será o punto para seguir polo robot cando se realicen os exemplos de seguimento de caras ou obxectos con diferentes partes do seu corpo, tal e como se mostra de novo na figura 6.2.2.4, onde se pode ver xa aplicado no propio robot. Polo tanto, este será o punto crave á hora de realizar aplicacións conxuntas entre as librerías de detección de imaxe e de movemento, o obter o punto central (referencia) do obxecto ou cara que se pretenda seguir.

En resumo, foi posible tanto o recoñecemento de obxectos de cor como o de caras. No caso do recoñecemento de obxectos deseñouse e implementouse unha etapa previa de recoñecemento de cor, tendo en conta as posibilidades de cambios na luminosidade ambiente.

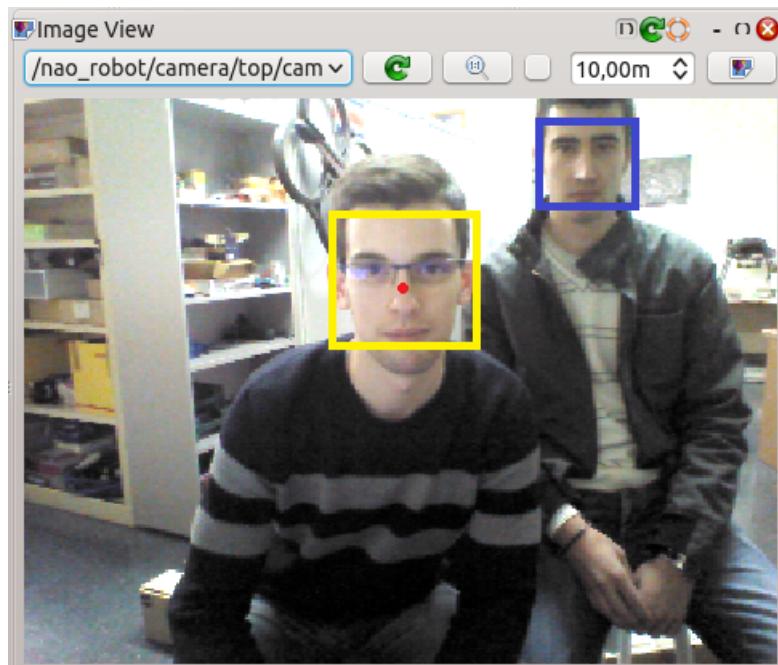


Figura 6.2.2.4 – Exemplo de detección cando se ven varias caras na imaxe

Xunto con isto, e para facer a detección máis robusta, optouse por deseñar un recoñecedor que, a parte de usar a cor do obxecto desexado, verifícase tamén a súa forma (neste caso circular) para poder evitar fallos de detección no caso de que algún obxecto da mesma cor se atopase cerca. Obtivéronse mellores resultados e más precisos que os que proporciona a API NAOqi dedicada ao seguimento de obxectos de cor, que tan só permite a posibilidade de seguir pelotas de cor vermello, pero non distingue a súa forma circular.

En canto ao recoñecemento de caras, deseñouse e implementouse un recoñecedor facial que puidera operar en ROS, xa que non se dispoña del. Neste caso fixose coa axuda das funcións da librería OpenCV dedicadas a tal fin, as cales foi necesario axustar e combinar de forma que se permitira realizar o recoñecemento en tempo real. Por outra parte, hai que comentar que o módulo realizado tan só distingue caras, e non persoas, polo que no caso de que existan diversas caras na imaxe, este optará por seguir a de maior tamaño, por considerarse a principal e más próxima ao robot.

Por último, tanto o módulo de recoñecemento de obxectos de cor, como o módulo de recoñecemento de caras, se deixaron preparados para o seu uso inmediato en ROS e documentados xuntos con diversos exemplos de aplicación, todos dispoñibles no enlace <https://github.com/naoTFM/Nao-Documetation-TFM>.

6.3. Librerías de son

Se a detección de imaxe é básica á hora de falar de HRI, non é moito menos importante o son, o cal inclúe dende o recoñecemento de voz ou a detección de fontes de son próximas, ata a sintetización de voz por parte do propio robot. Todo isto pode axudar aos robots a responder a acontecementos e ordes humanas con maior precisión.

Nesta sección coméntanse as bases de funcionamento de cada unha das librerías básicas de son empregadas, así como o traballo realizado para a súa posta en marcha.

6.3.1. Recoñecemento de voz

Para o recoñecemento de voz emprégase principalmente o módulo **ALSpeechRecognition** de NAOqi, que lle da ao robot a capacidade de recoñecer palabras ou frases predefinidas en varios idiomas, neste caso os idiomas utilizados son o castelán e o inglés. A pesar de que nun primeiro momento se pensou en emplegar a librería PocketSphinx, pero acabou por descartarse debido á maior complexidade existente para a súa posta en funcionamento, e debido tamén a que dentro dos requisitos de deseño se atopa o de usar os diferentes módulos de NAOqi.

ALSpeechRecognition baséase en sofisticadas tecnoloxías de recoñecemento de voz proporcionadas por NUANCE.

O seu principio de operación pódese resumir nos seguintes pasos:

1. Antes de comezar, “ALSpeechRecognition” necesita ser alimentado pola lista de frases que deben ser recoñecidas.
2. Unha vez iniciado, “ALSpeechRecognition” coloca na clave “SpeechDetected”, un booleano que especifica se se oe ou non un falante.
3. Se se oe un falante, o elemento da lista que mellor se adapte ao que se oe polo robot colócase na clave “WordRecognized”, a cal se organiza da seguinte maneira:

$$[Frase_1, confianza_1, frase_2, confianza_2, \dots, frase_n, confianza_n]$$

onde:

$Frase_i$ é unha das frases predefinidas e

$Confianza_i$ unha estimación da probabilidade de que esta frase sexa realmente o que foi pronunciado polo falante humano.

Observar que as diferentes hipóteses contidas nesa clave están ordenadas para que as frases más probables veñan primeiro, polo tanto bastará con acceder á primeira posición deste vector para coñecer a palabra ou frase que escoitou Nao.

6.3.2. Sintetizador de voz

O módulo **ALTextToSpeech** permite ao robot falar a través do envío de comandos a un motor de conversión de texto a voz e a posibilidade de personalización desa voz. O resultado da síntese será enviado aos altofalantes de Nao para que desta forma o poida escutar o usuario que estea a interactuar con el.

O sintetizador de voz está proporcionado polo Grupo ACAPELA, e tamén ofrece a posibilidade de adaptar os xestos e a fala do robot en función do contexto; aínda que no caso das probas realizadas neste traballo tampouco ten gran relevancia.

6.3.3. Localización da fonte de son

Como xa se veu dicindo, un dos propósitos principais de ter un robot humanoide é telo interactuando coa xente. Esta é sen dúbida unha tarefa difícil que implica unha boa cantidade de características. Ser capaz de entender o que se di e responder en consecuencia é certamente crítico pero en moitas situacións, estas tarefas requirirán que o robot estea primeiro na posición apropiada para sacar o máximo proveito dos seus sensores e deixar que a persoa considerada saiba que o robot está realmente escutando/falando con ela; isto poderá mostralo orientando a cabeza na dirección pertinente.

A función **ALSoundLocalization** aborda este problema ao identificar a dirección de calquera son ‘suficientemente alto’ oído por Nao.

A localización da fonte de son investigouse durante moito tempo e propuxéronse un gran número de enfoques. Estes métodos baséanse nos mesmos principios básicos, pero teñen un comportamento diferente e requieren diferentes cargas de CPU. Para producir saídas robustas e útiles, cumplindo cos requisitos de CPU e memoria do noso robot, a función de localización de fontes de son de Nao baséase nun enfoque coñecido como “Time Difference of Arrival”.

A onda sonora emitida por unha fonte próxima a Nao recíbese en momentos lixeiramente diferentes en cada un dos seus catro micrófonos. Por exemplo, se alguén fala co robot no seu lado esquerdo, o sinal correspondente golpeará primeiro os micrófonos esquerdos, uns poucos milisegundos más tarde os dianteiros e traseiros e finalmente o sinal será detectada no micrófono derecho.

Na figura 6.3.3.1 móstrase unha vista esquemática da dependencia entre a posición da fonte de son (un ser humano neste exemplo) e as diferentes distancias que a onda sonora necesita percorrer para alcanzar os catro micrófonos de Nao.

Estas diferenzas, coñecidas como TDOA (Time Differences Of Arrival), están relacionadas coa localización actual da fonte emisora. A partir desta relación cada vez que se oe un ruído,

o robot é capaz de recuperar a dirección da fonte emisora (ángulos azimutais e de elevación) dos TDOA medidos nos diferentes pares de micrófonos.

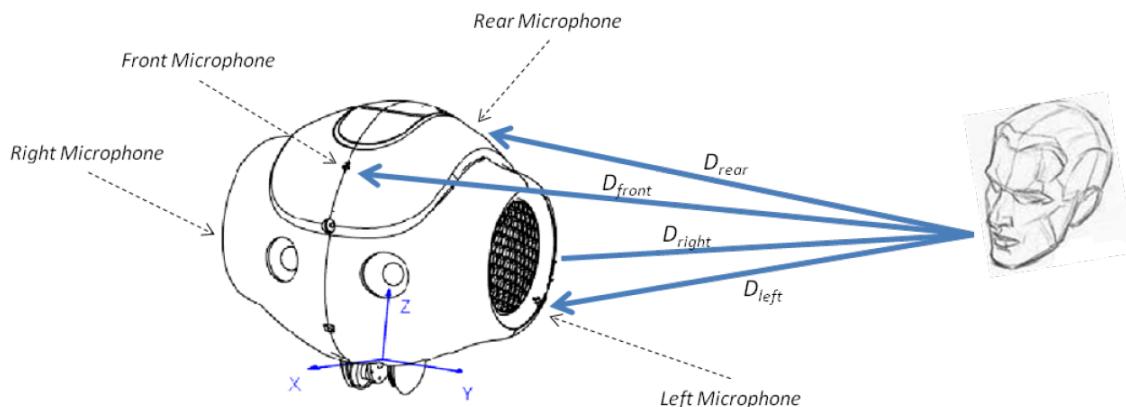


Figura 6.3.3.1 – Dependencia entre a posición da fonte de son e as distancias que a onda sonora percorre para alcanzar os micrófonos de Nao

Este método tamén presenta algunas **limitacións**, que se relatan a continuación:

- O rendemento da localización da fonte de son de Nao está limitado pola claridade con que se pode escutar a fonte de son con respecto ao ruído de fondo, xa que as contornas ruidosas tenden a diminuír a fiabilidade das saídas do módulo.
- Tamén detectará e localizará calquera “son forte”, sen ser capaz de filtrar se a fonte de son provén ou non de seres humanos.
- Finalmente, só se pode localizar unha fonte de son á vez. O módulo pode comportarse dunha maneira menos fiable se Nao se confronta a varios ruídos fortes ao mesmo tempo. Nese caso, é probable que só emita a dirección da fonte máis forte.

Por outra banda, hai que mencionar que a localización do son se activa cando se detecta un son, utilizando o algoritmo de **ALSoundDetection**. Ademais, cada vez que se realiza unha detección, a súa localización calcúlase e publícase no evento “**ALMemory/ALSoundLocalization/SoundLocated**”.

Por outra parte, para a localización do son tamén se fai uso do módulo **AITracker**, que permite ao robot seguir diferentes obxectivos, entre os cales se atopa a posibilidade de seguir un son, utilizando diferentes medios, como cabeza, corpo, movemento, etc. Pois ben, este módulo basea o seu comportamento para seguir os sons no módulo “**ALSoundLocalization**” comentando con anterioridade, e desta forma permite tamén saber con exactitude cal é a localización no espacio do obxectivo con respecto ao robot.

AITracker pode rastrexar o obxectivo cos modos mostrados na táboa 6.3.3.1 e pode seguir os obxectivos recollidos na táboa 6.3.3.2.

Modo	O robot seguirá ...	Comentario
Cabeza	coa cabeza soamente (modo por defecto)	As dúas xuntas da cabeza contrólanse para seguir o obxectivo.
WholeBody	con todo o corpo e pernas fixas	O robot mantén o equilibrio de forma autónoma e adapta a súa postura para rastrexar o obxectivo.
Movemento	con movemento	O robot móvese para manter unha distancia definida co obxectivo.

Táboa 6.3.3.1 – Táboa cos modos de seguimento

Obxectivo	Parámetros	Comentario
RedBall	Diámetro da pelota (metros)	Utilízase para calcular a distancia entre o robot e a pelota.
Cara	Anchura da cara (metros)	Utilízase para calcular a distancia entre o robot e a cara.
Son	[Distancia, confianza]	Distancia utilizase para estimar a posición do son e a confianza para filtrar a localización do son.

Táboa 6.3.3.2 – Táboa cos posibles obxectivos de seguimento

Por último, coa axuda da función **getTargetPosition** será posible coñecer a posición en [x, y, z] do obxectivo activo no momento (o son buscado) no marco de referencia torso. Isto faise asumindo un tamaño medio do obxectivo, polo tanto pode non ser moi preciso.

Hai que comentar que este módulo foi probado e funciona sen problema la mayoría das veces, pero o seu comportamento varía en función da distancia á que está situado o usuario. Se esta distancia está mal é posible que se xire cara o lado equivocado ou que o robot perda momentaneamente a referencia. Por outra parte, debido a que responde a calquera tipo de son ‘forte’, haberá que ter coidado co tipo de ambiente no que se vai a traballar con el.

En resumo, no caso da librería de son, estudáronse diversas posibilidades e tamén o funcionamento dos diferentes módulos existentes en NAOqi, polo que finalmente se optaría. Neste caso non se fixo o deseño, senón que só se implementaron as diferentes funcionalidades requiridas, para o cal se realizaron diversas probas de axuste de parámetros. Estas librerías están disponíveis para o seu uso e documentadas a través dunha serie de exemplos de aplicación, todo recollidos no enlace <https://github.com/naoTFM/Nao-Documetation-TFM>.

6.4. Librería de movementos básicos

Nesta sección inclúese todo o referido aos movementos básicos do robot, tales como camiñar ou mover as súas extremidades, xa sexa a cabeza para seguir un determinado obxectivo ou os brazos para interactuar ou coller algo coas mans.

6.4.1. Movemento do corpo

Para o movemento do corpo de Nao haberá que ter en conta varios aspectos, como son o marco de referencia no que se definen as accións, as diferentes articulacións do robot e as súas limitacións de movemento, e os efectores ou cadeas encargados de realizar o movemento conxunto de varias articulacións.

Desta forma, coñecendo todos estes parámetros, os movementos poderán quedar totalmente definidos. Tan só bastará con utilizar o correspondente módulo de NAOqi ou de ROS para poder executar cada un dos comportamentos e realizar todos os movementos necesarios, dende xirar a cabeza cara os lados ata camiñar.

Enténdese do anterior que é necesaria a explicación dos diferentes marcos de referencia cos que conta o robot, así como a descripción da totalidade dos datos da cinemática do robot: articulacións das que dispón, cadeas e efectores, e a situación dos diversos motores no seu corpo.

6.4.1.1. Marcos de referencia (Frames)

Cando se crea un comando para un robot, é necesario prestar moita moita atención ao espazo utilizado para definir o comando, é dicir, non será o mesmo definilo tomando como referencia a posición actual do humanoide que definilo tomando como referencia a posición do seu tronco ou a posición inicial de partida na que comezou a traballar. Por tanto, un erro no espazo elixido podería conducir a resultados desastrosos.

Os posibles marcos de referencia cos que conta Nao son os seguintes:

- **FRAME_TORSO.** Este marco adxúntase á referencia do torso do robot, polo que se move co robot mentres camiña e cambia de orientación a medida que Nao se inclina. Este espazo é útil cando se trata de tarefas moi locais, é dicir, nas que ten sentido utilizar como referencia o torso do robot.
- **FRAME_ROBOT.** Aquí a referencia é a media das posicións dos dous pés proxectadas ao redor dun eixo vertical Z. Este espazo é útil, porque o eixo X está sempre cara adiante, polo que proporciona unha referencia egocéntrica natural.

- **FRAME_WORLD.** Trátase dunha orixe fixa que nunca se altera. Déixase atrás cando o robot camiña, e será diferente en rotación respecto de Z despois de que o robot se vire. Este espazo é útil para cálculos que requieren un marco de referencia externo e absoluto.

Para maior claridade, poden verse estes marcos de referencia nas figuras 6.4.1.1 e 6.4.1.2.

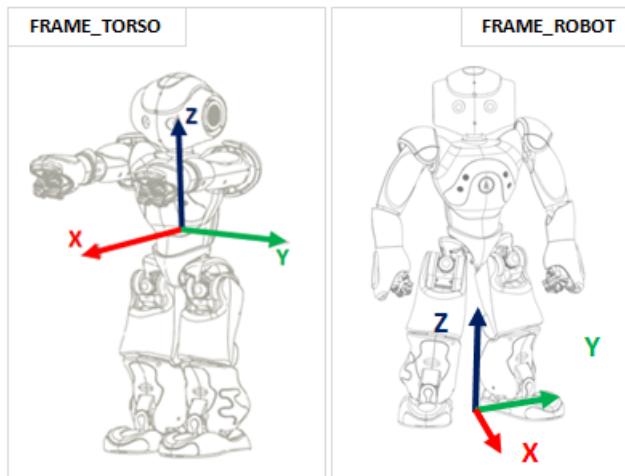


Figura 6.4.1.1 – Marcos de referencia Torso e Robot

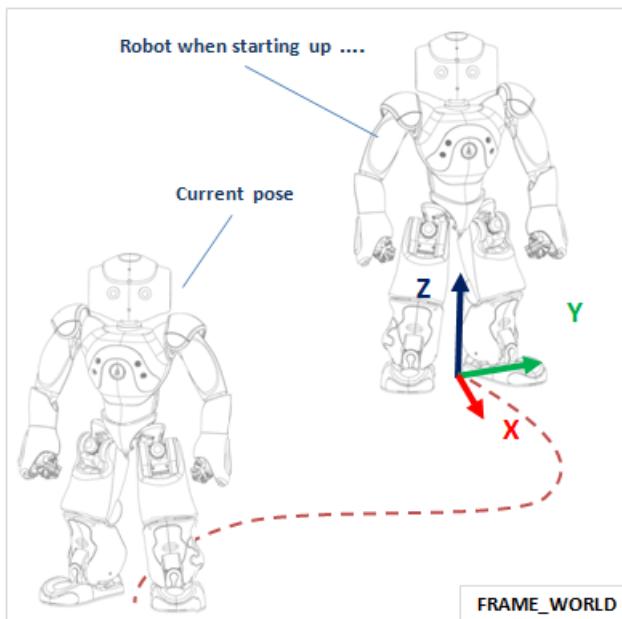


Figura 6.4.1.2 – Marco de referencia World

Por outra banda, ao executar unha tarefa, o espazo determiníñase no mesmo momento en que comeza a tarefa, e mantense constante durante o resto da interpolación. É dicir, a interpolación, unha vez definida, non cambiará se cambia a referencia (esta pode facelo debido a que as pernas se moven ou a cambios na orientación do torso).

6.4.1.2. Articulacións, efectores e cadeas

O robot Nao conta con 26 articulacións repartidas entre cabeza, brazos, pelvis e pernas. Cada unha das cales se pode controlar directamente e de forma individual ou en paralelo xunto a outras articulacións, a través das denominadas cadeas ou grupos de articulacións.

Para controlar unha articulación bastará con especificar o nome da articulación, o ángulo obxectivo en radiáns e a rapidez coa que se desexa ir a dito ángulo de destino.

A continuación indícanse as posicións de cada unha das diferentes articulacións existentes e os seus rangos de movemento, aínda que en primeiro lugar faise unha pequena aclaración acerca da convención de signos utilizada para definir cada un dos movementos destas sobre os eixos de coordenadas.

6.4.1.2.1. Convención de signos Dada unha articulación que une dúas partes do corpo do robot, a parte do corpo que está máis preto do tronco considérase fixa e a parte do corpo que está máis afastada do tronco é a que vira ao redor do eixo da articulación.

Para realizar a rotación das partes do corpo, colócase un marco de referencia en cada articulación. Cando o robot está en posición cero, todos os marcos de unión teñen a mesma orientación. Entón, as rotacións do rodete (Roll) teñen lugar ao redor do eixo X, as rotacións de cabeceo (Pitch) ao redor do eixo Y e as rotacións de guiñada (Yaw) ao redor do eixo Z, tal e como se mostra na figura 6.4.1.3.

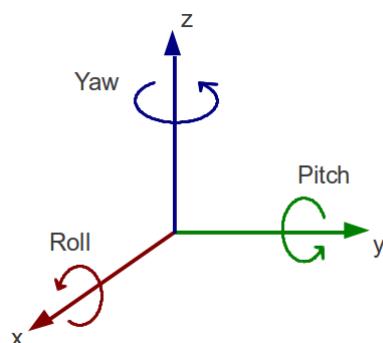


Figura 6.4.1.3 – Convención de signos (Roll, Pitch, Yaw)

6.4.1.2.2. Articulacións da cabeza Na cabeza atópanse dúas articulacións que serán moi útiles no transcurso das diferentes probas, xa que son as que permitirán seguir obxectos, caras ou sons a través do movemento da cabeza, estas son HeadPitch e HeadYaw, e atópanse representadas na figura 6.4.1.4.

O seu rango de movemento, mostrado na táboa 6.4.1.1, condiciona tamén o ángulo de visión co que conta o robot, debido a que a cámara empregada se atopa situada na cabeza.

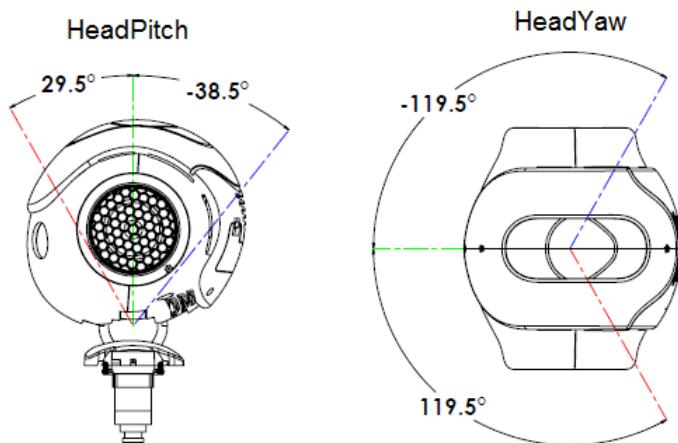


Figura 6.4.1.4 – Articulacións da cabeza

Nome da articulación	Movemento	Rango (graos)	Rango (radiáns)
HeadYaw	Torsión da articulación da cabeza (Z)	-119.5 a 119.5	-2.0857 a 2.0857
HeadPitch	Articulación da cabeza (Y)	-38.5 a 29.5	-0.6720 a 0.5149

Táboa 6.4.1.1 – Rango de movemento das articulacións da cabeza

Por outra parte, e debido a que existe a posibilidade de colisión entre cabeza e ombros, o rango de movemento Pitch está limitado segundo o valor de Yaw, é dicir, os movementos teñen unha limitación anti-colisión. No caso de empregar os módulos de NAOqi esta restrición vén establecida por defecto, e no caso de empregar ROS para levar a cabo os movementos, haberá que establecela manualmente mediante código.

6.4.1.2.3. Articulacións dos brazos Existen 6 articulacións en cada brazo repartidas entre ombreiro, cóbado, boneca e man. Estas articulacións serán de suma importancia á hora de manter o equilibrio para andar ou á hora da interacción co medio que rodea ao robot, por exemplo cando se trate de coller algúun obxecto.

Na figura 6.4.1.5 atópanse representadas as articulación correspondentes ao brazo esquerdo, para ao brazo derecho serán as mesmas pero cambiando a nomenclatura: RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, RWristYaw e RHand. A maiores, na táboa 6.4.1.4 atópanse recollidos os seus rangos de movemento.

6.4.1.2.4. Articulacións da pelvis Se os brazos eran importantes para camiñar e manter o equilibrio, sobra que dicir acerca da importancia da pelvis, que ademais é a encargada de permitir que o robot poida inclinar o seu tronco cara os lados e cara adiante ou atrás.

Neste caso, son dúas as articulacións existentes: LHipYawPitch e RHipYawPitch, áinda que

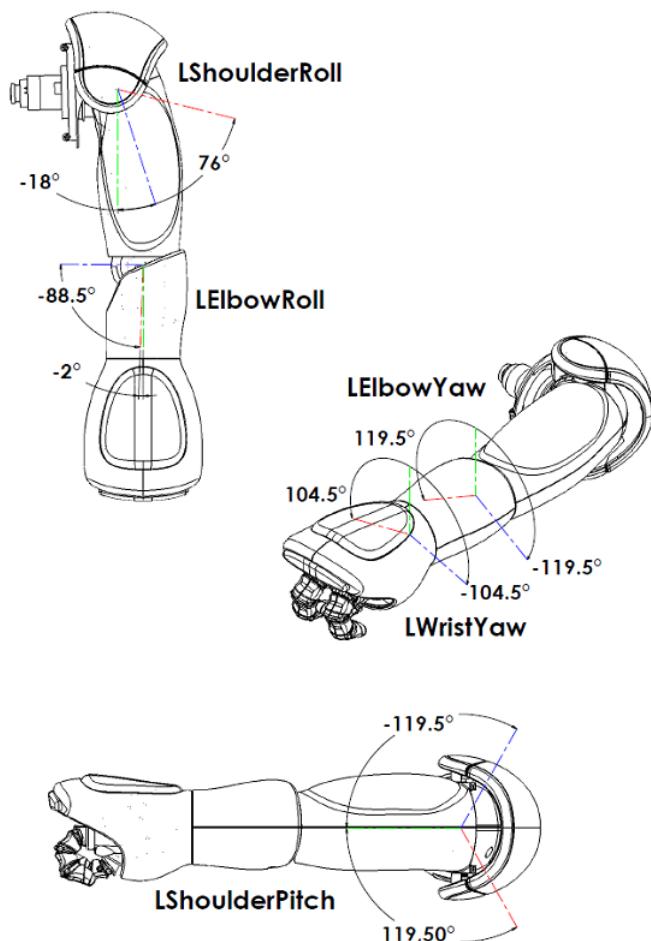


Figura 6.4.1.5 – Articulacións do brazo esquerdo

fisicamente son só un motor para que non poidan ser controladas de forma independente. De todas formas, en caso de ordes en conflito, LHipYawPitch sempre toma a prioridade. Na figura 6.4.1.6 móstranse ámbalas dúas articulacións, e na táboa 6.4.1.3 está recollido de forma máis concreta o seu rango de movemento.

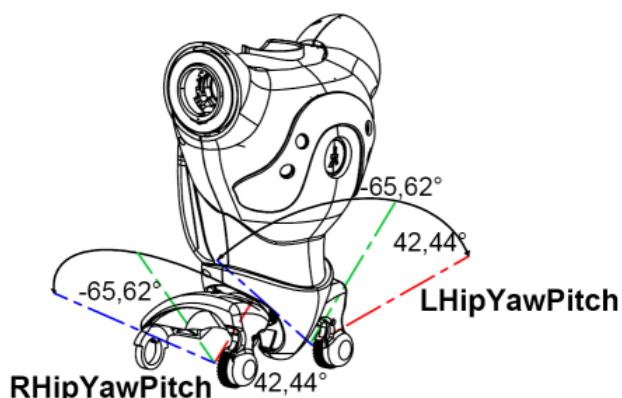


Figura 6.4.1.6 – Articulacións da pelvis

Nome da articulación	Movemento	Rango (graos)	Rango (radiáns)
LShoulderPitch	Articulación do ombreiro esquerdo (Y)	-119.5 a 119.5	-2.0857 a 2.0857
LShoulderRoll	Articulación do ombreiro esquerdo (Z)	-18 a 76	-0.3142 a 1.3265
LElbowYaw	Torsión da articulación do ombreiro esquerdo (X)	-119.5 a 119.5	-2.0857 a 2.0857
LElbowRoll	Articulación do cóbado esquerdo (Z)	-88.5 a -2	-1.5446 a -0.0349
LWristYaw	Articulación da boneca esquerda (X)	-104.5 a 104.5	-1.8238 a 1.8238
LHand	Man esquerda	Abrir e pechar	Abrir e pechar

Táboa 6.4.1.2 – Rango de movemento das articulacións do brazo esquierdo

Nome da articulación	Movemento	Rango (graos)	Rango (radiáns)
LHipYawPitch	Torsión da articulación da ca-deira esquerda (Y-Z 45°)	-65.62 a 42.44	-1.1453 a 0.7408
RHipYawPitch	Torsión da articulación da ca-deira dereita (Y-Z 45°)	-65.62 a 42.44	-1.1453 a 0.7408

Táboa 6.4.1.3 – Rango de movemento das articulacións da pelvis

6.4.1.2.5. Articulacións das pernas As últimas articulacións atópanse recollidas nas pernas do robot e son un total de 10, 5 en cada perna. Estas son as principais responsables de que o robot poida camiñar, manterse en pé, ou mesmo patear un balón.

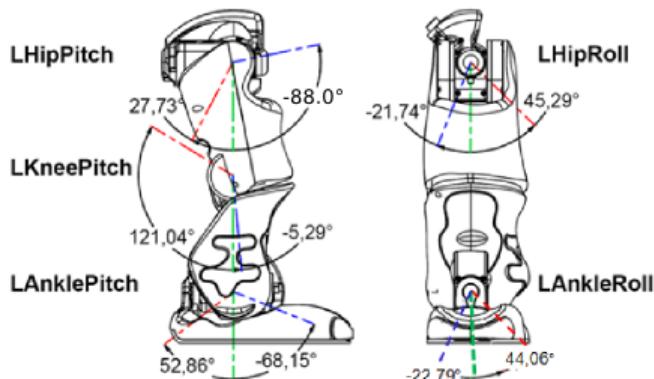


Figura 6.4.1.7 – Articulacións da perna esquerda

Do mesmo xeito que sucedía cos brazos, na figura 6.4.1.7 atópanse representadas as articulación correspondentes á perna esquerda, existindo unhas complementarias para a dereita. A maiores, na táboa 6.4.1.4 atópanse recollidos os seus rangos de movemento.

Por último, hai que comentar que debido á posible colisión de ámbalas pernas ao nivel do nocello, o rango de movemento de Roll está limitado de acordo co valor de Pitch. Neste caso, tanto se empregan os módulos de NAOqi como se opta por ROS, esta restrición vén establecida por defecto.

Nome da articulación	Movemento	Rango (graos)	Rango (radiáns)
LHipRoll	Articulación da cadeira esquerda (X)	-21.74 a 45.29	-0.3795 a 0.7905
LHipPitch	Articulación da cadeira esquerda (Y)	-88.00 a 27.73	-1.5359 a 0.4841
LKneePitch	Articulación do xeonlllo esquerdo (Y)	-5.29 a 121.04	-0.0923 a 2.1125
LAnklePitch	Articulación do nocello esquerdo (Y)	-68.15 a 52.86	-1.1895 a 0.9227
LAnkleRoll	Articulación do nocello esquerdo (X)	22.79 a 44.06	-0.3979 a 0.7690

Táboa 6.4.1.4 – Rango de movemento das articulacións da perna esquerda

6.4.1.2.6. Cadeas As cadeas non son outra cousa que agrupacións de varias articulacións baixo un mesmo nome, de forma que sexa posible o seu control conxunto.

Os diferentes grupos e cadeas existentes son os recollidos na táboa 6.4.1.5. A maiores, a cadea Body está formada por todas as cadeas xuntas, é dicir, comprende todas as articulacións de Nao.

Nome da cadea	Articulacións involucradas
Head	HeadYaw + HeadPitch
LArm	LShoulderPitch, LShoulderRoll, LEElbowYaw, LEElbowRoll, LWristYaw, LHand
LLeg	LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, RAnkleRoll
RLeg	RHipYawPitch, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, LAnkleRoll
RArm	RShoulderPitch, RShoulderRoll, REElbowYaw, REElbowRoll, RWristYaw, RHand

Táboa 6.4.1.5 – Articulacións pertencentes a cada cadea

6.4.1.2.7. Efectores Os efectores son puntos 3D predefinidos situados xeralmente ao final de cada cadea e que son de utilidade á hora de realizar cálculos relacionados coas cinemáticas directa e inversa. Desta forma, exercen o papel dunha referencia fixa que se debe situar nun punto determinado, e para o cal haberá que calcular as diferentes posicións intermedias das articulacións involucradas no movemento das correspondentes cadeas.

Os nomes dos distintos efectores son idénticos aos das cadeas, excepto para o caso de “Torso”. Na figura 6.4.1.8 é posible observar onde se atopan situados cada un deles sobre o corpo do robot.

A razón de uso destes puntos é porque cun robot bípedo pode quererse, entre outras cousas, controlar unha man co fin de atrapar un obxecto ou controlar un pé para patear unha pelota.

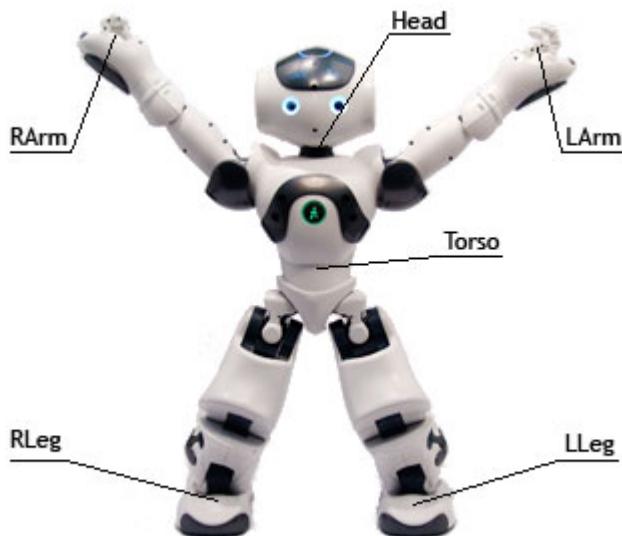


Figura 6.4.1.8 – Posición dos distintos efectores do robot Nao

6.4.1.3. Motores

Cada unha das articulacións de Nao atópase actuada por un motor, a excepción das articulacións da pelvis que, como ben se dixo antes, están actuadas polo mesmo. A figura 6.4.1.9 permite unha visión global da situación de cada unha das articulacións, cuxo movemento vai depender dos devanditos motores.

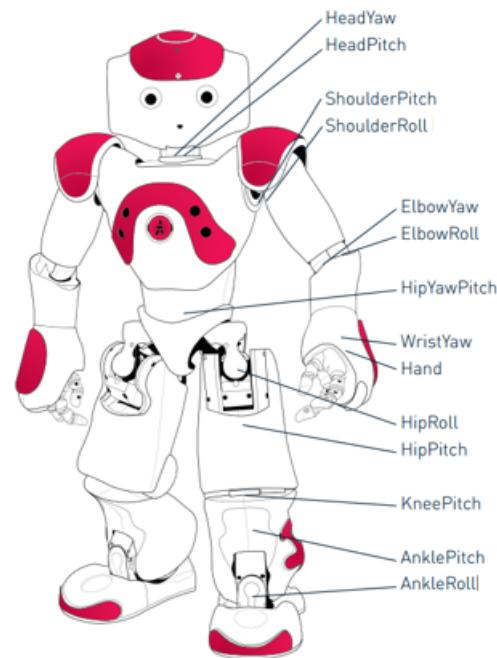


Figura 6.4.1.9 – Localización dos diferentes motores de Nao

Estes motores, á parte de permitir o movemento das diferentes articulacións, tamén fan posible controlar a rixidez (stiffness) de cada unha delas. Esta pode ser manexada de forma

global, en todo o robot, ou de forma específica, para unha ou varias articulacións. Por outra parte, o robot considerarase esperto cando pode pararse e moverse, é dicir, cando as articulacións das pernas están ríxidas.

A rixidez da articulación é equivalente a unha limitación de par nos motores. Se a rixidez da articulación está axustada a 0,0, o controlador da articulación non fai nada e esta está libre. Con todo, cun valor en 1,0, pódese utilizar a enerxía de par completo para alcanzar unha posición dada. Entre estes dous extremos, a articulación é máis ou menos ríxida e tenta alcanzar unha posición, pero se o par que ten que mover é maior que a limitación de rixidez, non se poderá alcanzar a posición obxectivo.

6.4.1.4. Desprazamento do robot

Neste apartado describense as funcións que se desenvolveron para mover ao robot na contorna.

No paquete *nao_driver* atópase un nodo denominado *nao_walker* no que xa está implementada a secuencia de movementos que permite ao robot andar sen caer, e os únicos parámetros que hai que configurar para que Nao se poida mover son as compoñentes lineal e angular da velocidade.

Para o desenvolvemento da librería fixose uso do tema */cmd_vel* , do tipo *geometry_msgs/Twist* que permite indicar ao robot as compoñentes de velocidad linear e angular para que este poida andar. O tipo de mensaxe */geometry_msgs/Twist* está formado por dous vectores de tres elementos tipo float cada un. O primeiro é o que indica os valores x, y, z da compoñente lineal da velocidad e o segundo os valores x, y, z da compoñente angular da velocidad.

O nodo *nao_walker* está subscrito a este tema e cada vez que un nodo publica unha actualización no mesmo, *nao_walker* encárgase de enviarlle a mensaxe actualizada ao robot para que este se movea dun xeito ou doutro. Esta estrutura móstrase na figura 6.4.1.10.

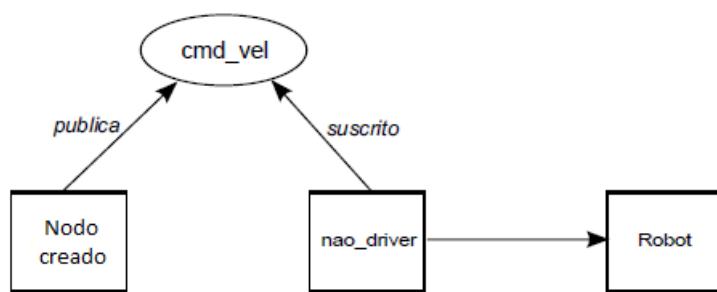


Figura 6.4.1.10 – Comunicación vía ROS para a librería de desprazamento

Unha vez descrito o uso do tema */cmd_vel* para conseguir o movemento do robot na súa contorna, pódese proceder á explicación de cada unha das librerías de movemento do robot

Nao para o movemento das súas articulacións.

6.4.1.5. Movemento das articulacións

Neste apartado explícase o funcionamento das funcións que permiten ao robot Nao mover as súas articulacións de forma independente. Estas funcións foron desenvolvidas facendo uso do nodo *nao_controller* do paquete *nao_driver* que se presentaba no apartado anterior.

O nodo *nao_controller* está subscrito ao tema */joint_angles*, que traballa con mensaxes do tipo *nao_msgs/JointAnglesWithSpeed*. Este tipo de mensaxes están formados polos seguintes campos:

- joint_names: Este campo de tipo string[] contén, separados por comas, os nomes das articulacións que se van a mover.
- joint_angles: Este campo é de tipo float[] e contén, no mesmo orde no que se introduciron os nomes das articulacións para mover no campo anterior, os ángulos que se desea mover cada unha respectivamente. Estes ángulos deben estar expresados en radiáns.
- speed: Este campo de tipo float indica a velocidade coa que se desea realizar o movemento da articulación. Toma valores entre 0 e 1, sendo 1 a máxima velocidade.

A comunicación entre as funcións e o robot segue o esquema presentado na figura 6.4.1.11.

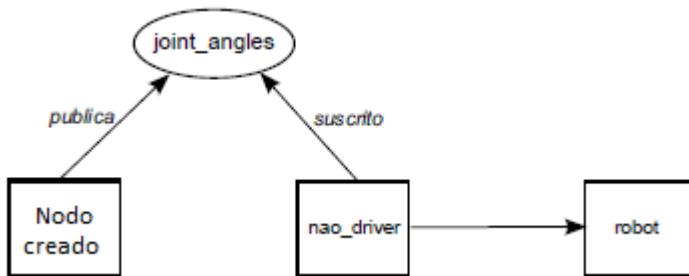


Figura 6.4.1.11 – Comunicación vía ROS para as funcións de movemento de articulacións

Anteriormente foron descritas as diferentes articulacións do robot e os seus correspondentes rangos de movemento, por iso cabe destacar que se o usuario executara unha función con un parámetro fóra do rango permitido, esta non chegará a executarse para evitar posibles danos físicos no robot.

6.4.2. Movemento de agarre de obxectos

O situar as mans correctamente para recoller un obxecto situado nun determinado lugar ten unha forte relación coa cinemática inversa: a configuración conxunta do brazo ou brazos

para alcanzar o obxecto debe ser tal que o manipulador remate preto das coordenadas correctas no espazo cartesiano. Aínda que este problema pode ser modelado e resolto mediante ecuacións, polo xeral hai máis dunha maneira de chegar á mesma posición e nesa situación debe tomarse unha decisión que sexa a mellor solución.

Nun primeiro momento, existían tres posibilidades á hora de realizar os cálculos necesarios para resolver este problema cinemático:

1. Usar a aplicación MoveIt! de ROS, que entre outras cousas permite a resolución da cinemática inversa.
2. Resolver a cinemática inversa á man.
3. Usar o módulo Cartesian Control de NAOqi.

Tras a realización de diversas probas para determinar cal das opcións é a mellor para satisfacelos requisitos deste traballo, chégase ás seguintes conclusóns:

- A aplicación MoveIt! está deseñada para poder xerar escenarios, evitar obstáculos ou introducir traxectorias moi complexas que sería imposible resolver de forma manual. Probose nun robot virtual e comprobáronse as distintas posibilidades que ten, pero á hora de utilizala nun robot real é necesario xerar un escenario da contorna do robot, con todos e cada un dos obxectos existentes nel, tarefa un tanto complexa para o que realmente se está a buscar.
- A opción de resolver a cinemática inversa á man descártase case de inmediato debido á falta de viabilidade á hora de resolver o problema para os infinitos casos existentes cando se pretende coller un obxecto.
- O módulo Cartesian Control de NAOqi incorpora un solver que selecciona a mellor opción das posibilidades existentes tras a realización da cinemática inversa e á súa vez realiza os cálculos con tan só indicarlle as coordenadas do obxecto respecto dun dos marcos de referencia do robot; por tanto, debido a que proporciona os mesmos resultados que MoveIt! e a súa forma de uso é relativamente máis sinxela, óptase por esta opción.

En resumo, para realizar esta librería de agarre de obxectos pártese das APIs de NAOqi dedicadas a resolver a cinemática inversa e contidas no módulo **Cartesian Control**. Polo tanto, as principais funcións a utilizar dentro deste módulo serán as seguintes:

- **positionInterpolations**, que move os efectores finais a unhas posicións e orientacións dadas no tempo e referidas a un dos marcos de referencia do robot.
- **getPosition**, que permite obter a posición relativa dunha articulación ou cadea, respecto ao marco de referencia desexado.

6.4.2.1. Cinemática inversa a través do módulo “Cartesian Control”

Estas APIs dedícanse a controlar directamente os efectores do robot nun espazo cartesiano utilizando un solver de cinemática inversa, podendo controlarse cada efector individualmente ou en paralelo con outros.

Hai dous tipos de solver de cinemática inversa (IK) no módulo ALMotion:

- Un solver clásico IK que utiliza só as articulacións da cadea do efector para alcanzar un obxectivo. Este é o solver que se vai a utilizar.
- Un solver xeneralizado IK que utiliza todas as articulacións do robot para alcanzar un obxectivo.

O modelo xeométrico do robot proporciona a posición dun efector $X = [P_x, P_y, P_z, P_{wx}, P_{wy}, P_{wz}]$ relativa a un espazo absoluto en función de todas as posicións dunha articulación ($q = [q_1, \dots, q_n]$).

$$X = f(q) \quad (6.4.2.1)$$

O modelo cinemático directo é a derivada da ecuación 6.4.2.1 con respecto ao tempo:

$$\begin{aligned} \dot{X} &= \frac{\delta}{\delta t} f(q) \dot{q} \\ &= J(q) \dot{q} \end{aligned}$$

onde $J(q)$ tamén se denomina matriz xacobiana.

Neste caso, quérese controlar un efector e deducir a posición dunha articulación. Polo tanto, o que se busca é o modelo cinemático inverso:

$$\dot{q} = J^{-1} \dot{X} \quad (6.4.2.2)$$

En moitos casos, J non é invertible directamente (matriz non cadrada), casos nos que se usará a pseudoinversa de Moore-Penrose para resolver matematicamente o problema.

Por outra parte, hai que comentar que o solver empregado basea o seu comportamento na interpolación SE3, que se utiliza para todas as interpolacións que se definen no espazo cartesiano. Este método proporciona unha interpolación tipo ‘spline’ que permite ter en conta as velocidades iniciais e os puntos de paso, garantindo traxectorias suaves que respectan as restriccións de velocidade.

Tamén se terá que ter en conta que, se o movemento desexado é inviable, o robot pode perder o equilibrio e caer. É por iso que todos os movementos cartesianos deberían ser probados nun simulador antes de ser probados no robot.

6.4.2.2. Agarre de obxectos a diferentes alturas: Pasos a seguir

Baseándose no hardware de Nao e na arquitectura do robot, a librería de agarre de obxectos pode dividirse naturalmente en dúas partes: o recoñecemento de obxectos e un comportamento de nivel superior para agarralos. Pois ben, este comportamento de nivel superior pódese dividir en tres sub-comportamentos ou etapas:

1. O primeiro descobre onde se atopa o obxecto que se vai a agarrar, é dicir, consiste na súa localización no espazo 3D.
2. O segundo deles consiste na secuencia de movementos a realizar para recoller o obxecto, é dicir, consistirá en resolver o antes comentado problema de cinemática inversa.
3. O terceiro validará que o obxecto foi realmente recollido.

A única limitación adicional que se impón é que o obxecto debe estar centrado diante de Nao antes de tentar agarralo. Na práctica, isto non é unha limitación, porque o robot pode resolver este problema realizando unha rotación ou dando uns pasos cara ao lado para cambiar a súa posición relativa ao obxecto e que este último estea centrado diante do robot.

Agora ben, tendo en conta que vai ser praticamente imposible conseguir un agarre correcto o 100 % das veces, debido maioritariamente a que os sensores teñen unha marxe de erro e non teñen unha precisión exacta, tratarase de facer que a porcentaxe de acerto sexa a máis alta posible. Para conseguir isto haberá que cumplir unha serie de requisitos:

- **Distancia exacta entre o robot e a pelota.** Antes de proceder ao agarre do obxecto, haberá que achegarse a el ata atoparse a unha distancia na cal sexa posible collelo xusto cos brazos estirados. Será necesario determinar que radio tería o obxecto visto a través da cámara de Nao a esa distancia para saber cando parar.
- **Pelota centrada.** Á súa vez, ao atoparse á distancia indicada do obxecto débese proceder a situar cabeza e brazos correctamente rectos, asegurando tamén que o centro do obxecto debe coincidir na franxa central da cámara de Nao. Para iso, será necesario centrar a pelota na imaxe rotando o corpo do robot cara a ela. Por tanto, coñecido o centro da pelota e coñecido o centro da imaxe, se a coordenada X da pelota se atopa dentro dun rectángulo central, véxase a figura 6.4.2.1, dentro do cal se pode considerar que a mesma está centrada, a pelota estará no centro, noutro caso haberá que seguir desprazándose cara ao lado correspondente.

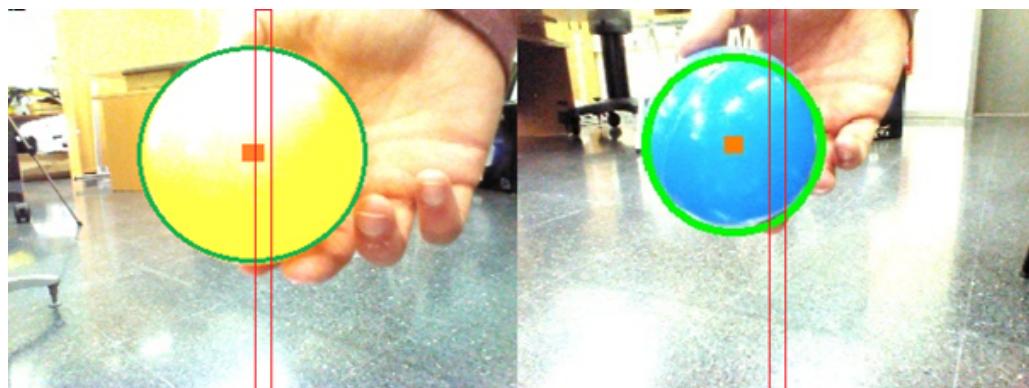


Figura 6.4.2.1 – Exemplos de posicionamento do centro da pelota respecto á franxa central da imaxe

- **Aliñación cabeza-corpo.** A cabeza debe estar recta e perfectamente aliñada co corpo do robot, así desta forma será posible coñecer a que altura realizar o agarre, en función da altura respecto ao centro da imaxe vista a través da cámara, para a cal se utilizará como marco de referencia o correspondente ao torso do robot. Será necesario comprobar o valor de HeadYaw para saber se a cabeza está ou non virada respecto ao corpo. Na figura 6.4.2.2 móstrase o exemplo de como o robot ao detectar que non ten a cabeza e o corpo aliñados, se move cara o lado para desta forma centrarse e situarse nunha posición máis óptima.

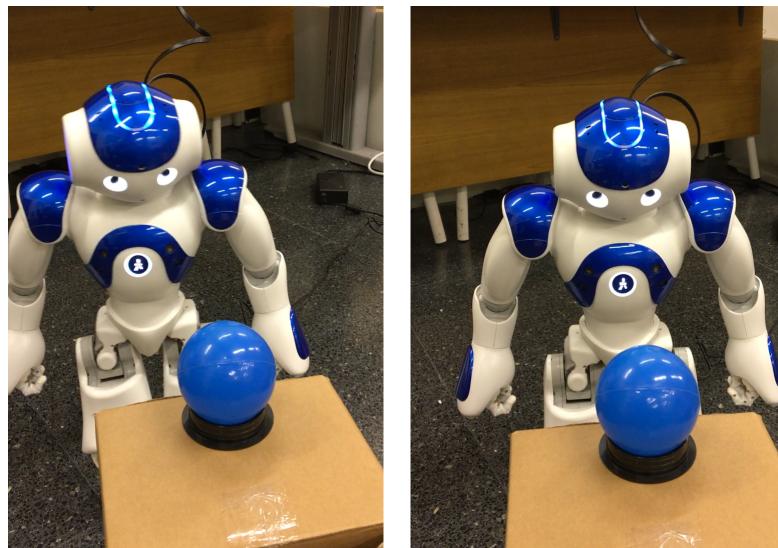


Figura 6.4.2.2 – Exemplos de aliñación da cabeza respecto ao corpo

Unha vez se cumpriron todos os requisitos e se estimou a posición da pelota no espazo 3D (que se explicará despois con más detalle), xa será posible levar a cabo o proceso de agarre. Este proceso, mostrado na figura 6.4.2.3 , divídese en tres etapas:

1. Colocar os brazos á dereita e esquerda do obxecto pero sen collelo, para comprobar que non existen errores de localización.
2. Mover os brazos o un cara ao outro ata agarrar o obxecto.

3. Finalmente mover os brazos co obxecto agarrado cara ao peito do robot para evitar chocar co lugar no que se encontraba o dispositivo colocado e que o robot sexa máis estable ao moverse.



Figura 6.4.2.3 – Secuencia de agarre de obxectos

6.4.2.3. Localización do obxecto no espazo 3D

Tal e como se comentou con anterioridade, a resolución de cinemática inversa baseárase na configuración conxunta dos brazos para alcanzar o obxecto, de forma que os manipuladores rematen preto das coordenadas correctas no espazo cartesiano. Debido a isto, é necesario facer unha breve explicación acerca de como se van a calcular estas coordenadas do obxecto respecto ao robot.

Primeiro de nada, debe quedar claro que para a localización eficaz do obxecto, neste caso unha pelota, no espazo 3D, o seu centro e o seu radio son coñecidos e serán proporcionados pola librería de detección de obxectos. Por tanto, haberá que converter esta posición a coordenadas do mundo real co robot como orige, onde o eixo X é positivo cara á parte dianteira de Nao, o Y de dereita a esquerda e o Z é vertical; é dicir, o marco de referencia será o torso do robot, tal e como se podía apreciar na figura 6.4.1.1.

6.4.2.3.1. Coordenada X Para o cálculo da coordenada X tómase unha imaxe da pelota a unha distancia de z centímetros fixa e coñecida, e calcúlase o radio da mesma sobre esa imaxe, que serán r píxeles. Desta forma poderase usar un conversor ‘radiusToMeters’ que terá a forma:

$$\text{radiusToMeters} = z \cdot r(\text{cm} \cdot \text{px})$$

Por tanto, para coñecer a distancia á que se atopa a pelota do robot (coordenada X) bastará con coñecer o radio da pelota e dividilo por este conversor ‘radiusToMeters’. Na figura 6.4.2.4 móstrase de forma esquemática a que se refire esta coordenada X, marcada como ‘Xrobot’ para que non sexa confundida coa coordenada ‘x’ do centro da pelota.

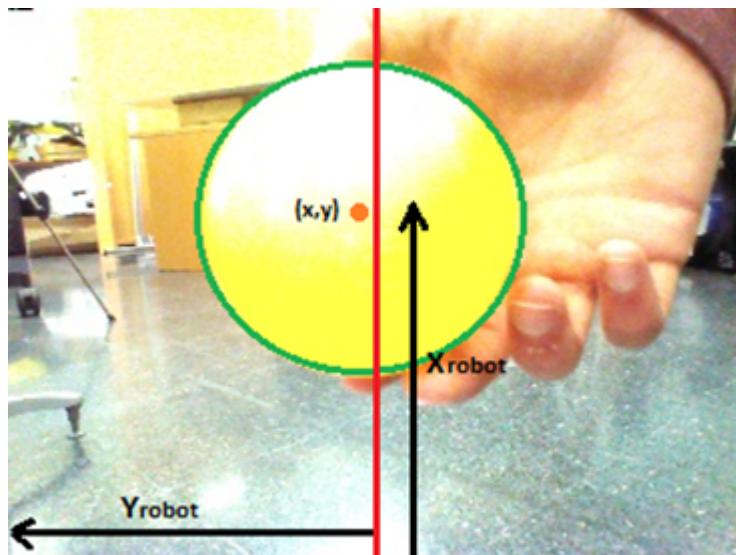


Figura 6.4.2.4 – Coordenadas X e Y respecto ao marco de referencia do torso do robot

No caso dos exemplos realizados neste traballo, a imaxe foi tomada a unha distancia $z = 25\text{cm}$ obténdose un radio da pelota de $r = 55\text{px}$ sobre dita imaxe. Valores a partir dos cales foi calculado o conversor ‘radiusToMeters’.

6.4.2.3.2. Coordenada Y Para a coordenada Y existen 2 posibilidades: que a pelota se atope á esquerda do centro, entón a coordenada Y é positiva, ou que se atope á dereita, sendo neste caso negativa.

A pelota ten un diámetro real de D milímetros. Con este valor tamén se pode crear un conversor ‘pixelToMeter’ que se obtén dividindo este valor do diámetro real entre o diámetro d da pelota en píxeles, de forma que é posible coñecer a distancia dun píxel en metros.

Por outra banda, para coñecer en que lado se atopa situada a pelota bastará con restarlle ao valor en píxeles do centro da imaxe (neste caso 160 porque a imaxe ten un ancho de 320 píxeles) o valor ‘x’ do centro da pelota tamén en píxeles. Desta forma obterase unha coordenada Y positiva se ‘x’ é máis pequeno que 160 e por conseguinte se atopa á esquerda do centro da imaxe, e unha coordenada Y negativa se o centro da pelota se atopa á dereita da imaxe. Finalmente multiplícase o valor obtido para a coordenada Y en píxeles polo conversor ‘pixelToMeter’ para coñecer a que distancia en metros se encontra situada a pelota.

Do mesmo xeito que antes, na figura 6.4.2.4 móstrase de forma esquemática a que se refi-

re esta coordenada Y, marcada como 'Yrobot' para que non sexa confundida coa coordenada 'y' do centro da búa.

No caso dos exemplos realizados neste traballo, a pelota empregada ten un diámetro $D = 8\text{mm}$. Valor co que se realizou o cálculo do conversor 'pixelToMeter'.

6.4.2.3.3. Coordenada Z A altura á que se atopa a pelota será estimada en función da inclinación que presente a cabeza do robot ao ver a pelota xusto no centro da imaxe, desta forma poderíase medir o seo do ángulo que forma a inclinación da cabeza (para ver a pelota) coa horizontal, da forma mostrada na figura 6.4.2.5, ou a distancia equivalente entre o torso do robot e a posición da pelota.

Debe existir, polo tanto, un paso previo a esta estimación no cal o robot axuste a inclinación da súa cabeza de forma que o centro da pelota e o da imaxe vista a través da cámara coincidan. Esta inclinación vén dada polo valor da articulación HeadPitch e o seu valor varía da forma en que se vía na figura 6.4.1.4.

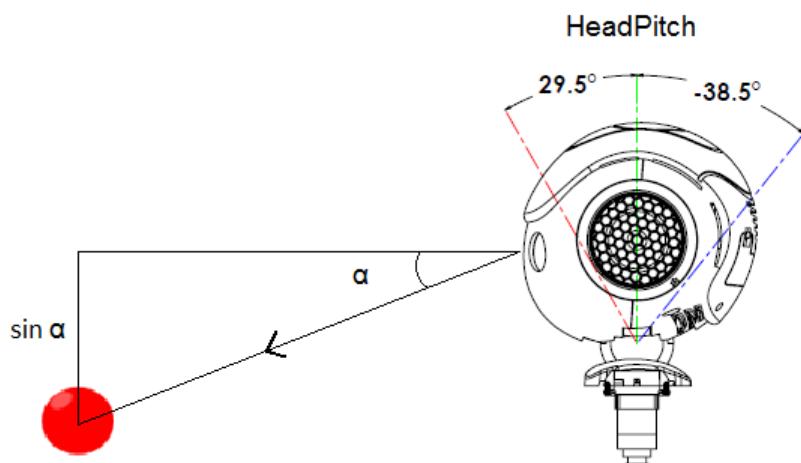


Figura 6.4.2.5 – Altura da pelota en función do ángulo entre a cabeza e a horizontal

Así, a partir deste valor de inclinación pódese comprobar que existe unha relación lineal coa altura á que se atopa a pelota. Calcularase por tanto a ecuación que rexe esta relación lineal para que a partir dun valor dado de HeadPitch se obteña de forma automática a altura ou coordenada Z. Hai que ter en conta que esta distancia será a medida con respecto ao eixo de coordenadas que se atopa no torso do robot, e non con respecto ao chan.

Na táboa 6.4.2.1 móstranse os resultados obtido tras a realización de diversas medicións para tratar de obter a relación lineal entre a altura á que se atopa o obxecto e a inclinación da cabeza do robot, dada polo valor da articulación HeadPitch. En dita táboa móstranse os diferentes valores medios de dita articulación para a altura determinada. Por outra parte, comentar

que a columna 'Altura real' se refire á altura á que se atopa o obxecto respecto ao chan e a columna 'Altura Nao' refírese de novo á altura pero respecto ao marco de referencia Torso do Robot, que será o empregado despois para a realización dos movementos de agarre.

Altura real (cm)	Altura Nao (cm)	HeadPitch (rad)
32	11.5	0.454
35	14.5	0.332
38	17.5	0.222
40.5	20	0.141
43.2	22.7	0.021

Táboa 6.4.2.1 – Valores de HeadPitch en función da altura á que está situado o obxecto

Unha vez coñecidos todos os valores, cómpre representalos nun gráfico para ver que tipo de relación existe entre eles. Na figura 6.4.2.6 pode verse que se trata dunha relación lineal, polo que bastará con calcular a ecuación da recta que relaciona a altura á que está a pelota coa inclinación da cabeza do robot, para coñecer unha en función do valor da outra.

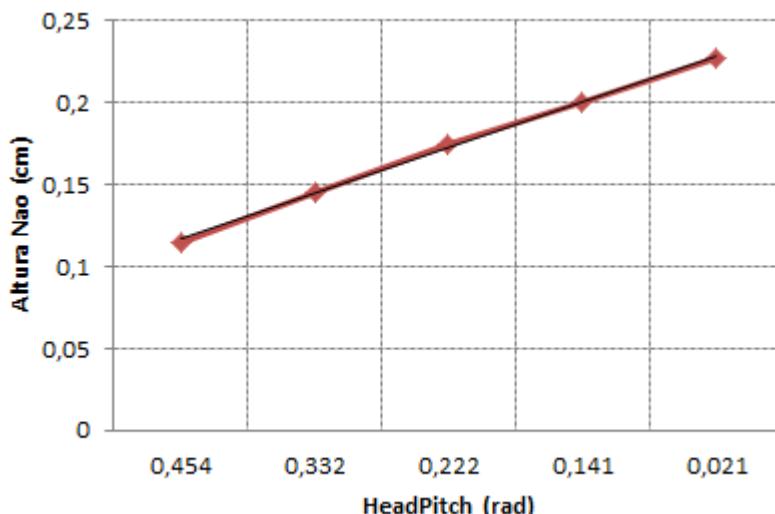


Figura 6.4.2.6 – Gráfico coa altura da pelota respecto ao torso do robot e o valor de HeadPitch

Pois ben, coñecendo que se trata dunha recta, cuxa ecuación é a mostrada na ecuación 6.4.2.3, será posible coñecer o valor da coordenada Z do obxecto con tan só fixarse no valor de HeadPitch.

$$Z = (-25,866 \cdot \text{HeadPitch} + 23,843)/100 \text{ (m)} \quad (6.4.2.3)$$

Por último, unha vez que é posible situar o obxecto no espazo 3D, xa que todas as súas coordenadas cartesianas son coñecidas, estas serán publicadas como un Punto nun tema de ROS, para que o módulo correspondente ao agarre poida traballar con elas.

Nas figura 6.4.2.7 pódense ver distintos exemplos de agarres realizados a diferentes alturas, de forma que se comproba que a relación calculada con anterioridade para a determinación da coordenada Z é correcta, o que permite a obtención duns resultados satisfactorios.



Figura 6.4.2.7 – Exemplo de agarres a diferentes alturas

Cabe comentar que, de todas as coordenadas publicadas no punto, no exemplo que se vai a realizar neste traballo tan só se usará a correspondente á altura, a coordenada Z. As outras non será necesario facer uso delas, xa que sempre se collerá a pelota cando esta se atope diante do robot e este teña o seu corpo correctamente orientado cara a ela. Aínda así, os valores de X e Y son publicados igual porque son útiles para o futuro, para calquera outra aplicación de agarre ou de comunicación con outro robot, por exemplo.

A modo de resumo final, foron postos a disposición do usuario en ROS diferentes módulos que permiten ao robot camiñar ou mover calquera das súas articulacións. Á súa vez, e debido a que para realizar o agarre de obxectos é preciso coñecer de antemán a súa situación no espazo, foi deseñado e implementado con éxito un algoritmo que permite estimar dita situación no espazo 3D con respecto ao marco de referencia situado no torso do robot, que será o que se empregue para calcular os movementos necesarios para levar a cabo a captura con éxito.

Empregáronse uns conversores para calcular as coordenadas X e Y e buscouse outra relación para poder estimar a altura á que se atopa o obxecto (coordenada Z). Tamén hai que comentar que esta última estimación foi bastante laboriosa, debido a que se partiu de varias ideas que en principio parecían correctas, pero que acababan fallando nalgún punto, ata que se chegou á solución final na que se viu que era posible calcular a altura do obxecto en función da inclinación da cabeza para velo, sempre e cando o robot estivese parado á mesma distancia de dito obxecto.

Por último, tanto o módulo de movemento do corpo, como o módulo de agarre de obxectos, se deixaron preparados para o seu uso inmediato en ROS e documentados xuntos con diversos exemplos de aplicación, todo dispoñibles no enlace <https://github.com/naoTFM/Nao-Documentation-TFM>.

7 Probas e resultados

Neste capítulo expoñeranxe os resultados e o funcionamento das diferentes librerías desenvolvidas neste traballo mediante unha serie de aplicacións de exemplo.

Cabe mencionar tamén que todos os códigos dos exemplos e as librerías realizados están recollidos en github (<https://github.com/naoTFM/Nao-Documentation-TFM>) para a súa consulta pública, así como a documentación de cada unha delas más os vídeos coas diferentes simulacións realizadas.

7.1. Contorna das probas

Os experimentos realizáronse no laboratorio GOI do Edificio de Talleres Tecnolóxicos da Universidade Da Coruña, no Campus de Esteiro. Na figura 7.1.0.1 móstranse os elementos que componen a contorna das probas.



Figura 7.1.0.1 – Elementos para a realización das diferentes probas

Os experimentos foron calibrados para realizar as probas no laboratorio, onde non existen cambios de luminosidade na contorna, debido a que soamente está iluminado por medio da luz artificial arroxada por unha serie de lámpadas distribuídas pola habitación.

Ao longo do desenvolvemento do traballo, segundo se ían implementando novas librerías para as distintas funcionalidades do robot, fóreronse probando de forma individual para comprobar que o funcionamento das mesmas era o esperado. Estas probas de carácter sinxelo serviron para comprobar que as funcións codificadas cumpren os requisitos que se especificaron no capítulo 5.

A continuación detállanse distintos experimentos que simulan comportamentos avanzados que poden realizarse mediante combinacións das diferentes librerías que foron desenvolvidas.

7.2. Seguimento dun obxecto de cor coa cabeza ou con todo o corpo

Este exemplo permite ao robot Nao ser capaz de seguir unha pelota de cor coa cabeza e moverse cara a ela ata atoparse a unha determinada distancia da mesma, para o que serán necesarias as librerías de detección de imaxe e de movemento.

Previo ao seguimento da pelota conta cunha etapa de recoñecemento e/ou calibrado da cor do obxecto, de forma que o rango de detección desexado se calcula de forma automática sen ter que variar ningún parámetro nas librerías.

Por tanto, o funcionamento xeral deste exemplo pódese dividir en dúas etapas:

1. Detección do rango de cor HSV.
2. Seguimento da pelota coa cabeza ou con todo o corpo.

A primeira etapa de detección do rango de cor realiza-se coa axuda dos botóns táctiles incluídos na parte superior da cabeza do robot Nao, que son os mostrados na figura 7.2.0.1.

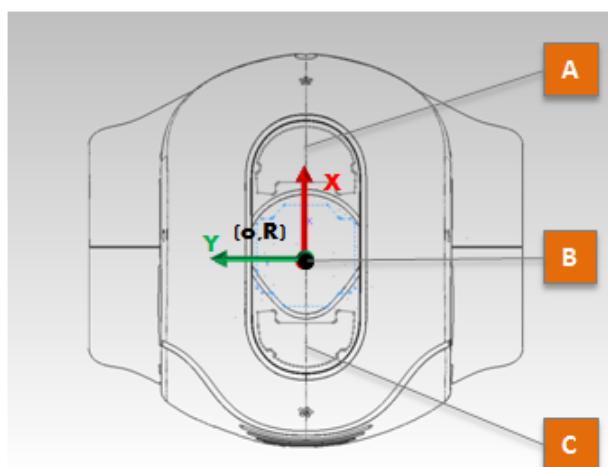


Figura 7.2.0.1 – Botóns táctiles situados na cabeza de Nao

O modo de funcionamento dos devanditos botóns táctiles atópase recollido en forma de diagrama na figura 7.2.0.4 e é o seguinte:

- **Botón A: Frontal.** Ao pulsar este botón desactívase o modo “tracking”, por tanto o robot deixa de seguir a pelota. Á súa vez coloca os seus brazos en posición de recoñecemento, é dicir, estirados cara adiante, para que a pelota sexa colocada entre as súas mans e asegurar desta forma que se vexa perfectamente a forma circular da mesma e que así sexa más precisa a detección da súa cor, tal e como se pode ver na figura 7.2.0.2.



Figura 7.2.0.2 – Detección da forma circular da pelota na posición de recoñecemento

- **Botón B: Medio.** Ao pulsar este botón, e unha vez colocada a pelota entre as mans de Nao para que este poida vela, realiza unha captura da imaxe vista a través da cámara superior do robot, de forma que nesta imaxe se poida recoñecer a pelota e facer unha estimación da cor HSV dos píxeles da mesma, tal e como se mostra na figura 7.2.0.3.



Figura 7.2.0.3 – Detección dos píxeles más representativos para a estimación

As accións realizadas ao pulsar este botón pódense resumir da seguinte forma:

1. Obtención de imaxe a partir da cámara superior de Nao.
2. Delimitación do contorno e do centro da pelota para delimitar en que zona coller os píxeles.
3. Cálculo da media HSV dos píxeles elixidos.

4. Determinación do rango de detección a partir do valor HSV calculado.

- **Botón C: Traseiro.** Coa pulsación deste terceiro botón dáse por concluída a fase de recoñecemento de cor e comézase co seguimento da pelota de cor identificada. Á súa vez, o robot pon os seus brazos de novo na posición inicial (estirados á beira do seu corpo) para facer ver así que xa comeza coa etapa de seguimento.

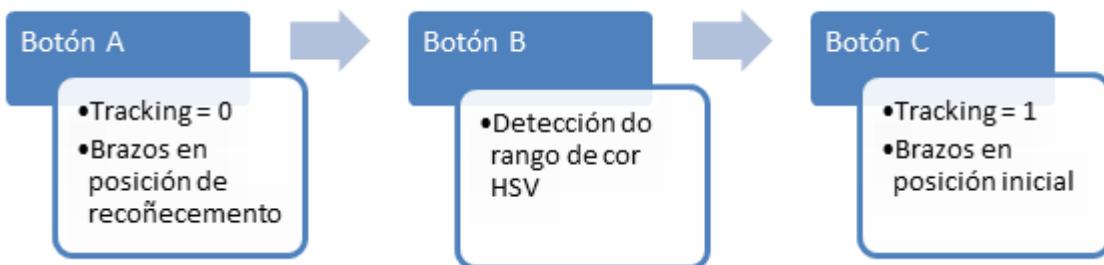


Figura 7.2.0.4 – Modo de funcionamiento dos botóns táctiles

En canto á etapa de seguimento da pelota, o primeiro que haberá que ter en conta é o movemento de cabeza necesario para realizarlo. Este calcularase de forma que se manteña sempre a pelota no centro da imaxe que se está obtendo en tempo real a través da cámara superior do robot. Na figura 7.2.0.5 móstrase este cálculo explicado e recollido dentro dun rectángulo de cor verde.

```

# Obtención do centro do obxecto detectado e cálculo dos xiros da cabeza
target_x =center[0] # coordenada x do centroide
target_y=center[1] # coordenada y do centroide

# Centro da imaxe -> (cv_image.shape[1]/2, cv_image.shape[0]/2)=(320/2,240/2)   (x,y) coordenadas
sub_x = target_x - cv_image.shape[1]/2
sub_y = target_y - cv_image.shape[0]/2

var_x = (sub_x / float(cv_image.shape[1]/2))
var_y = (sub_y / float(cv_image.shape[0]/2))

# Comproba a posición da cabeza antes de move-la para restrinxir os movementos e evitar que se choque cos ombreiros

# Determino a nova posición relativa en Y, que é a que se restrinxen para evitar choques
new_position_y=self.positions[1]+var_y*0.10

if (new_position_y <= -0.45 ):
    var_y=0
elif (new_position_y >= 0.3):
    var_y=0

joint.joint_angles.append(-var_x*0.25)
joint.joint_angles.append(var_y*0.10)

self.joint_pub.publish(joint)
  
```

Figura 7.2.0.5 – Exemplo do código empregado para calcular o movemento da cabeza e a súa restrición

Como se comentou en capítulos anteriores, algúns movementos da cabeza atópanse restrinxidos, para evitar que esta choque cos ombreiros do robot e evitar así o seu desgaste. Para iso, en cada iteración realiza unha predición da que será a nova posición da cabeza e, se esta é conflitiva, restrínxese o seu movemento para evitar o conflito. Na figura 7.2.0.5 atópase

recollido este cálculo dentro dun cadeo de cor laranxa.

No que ao movemento do corpo cara á pelota respecta, este faise en función da distancia á que se atope dita pelota (usándose como parámetro característico para determinar isto o seu radio) e en función da posición da cabeza do robot, de forma que este se desprase cara adiante ou cara aos lados segundo a posición na que se atope o obxecto.

Tómase como radio de referencia para comezar a andar 50 milímetros, que é o correspondente ao da pelota vista a través da cámara cando está a unha distancia equivalente á lonxitude do brazo do robot estirado cara a adiante. A partir de aí, se a pelota se afasta, o radio faise menor e o robot camiña cara a ela ata volver quedar á mesma distancia, e se a pelota se achega, Nao non se moverá e tan só realizará o seguimento coa cabeza.

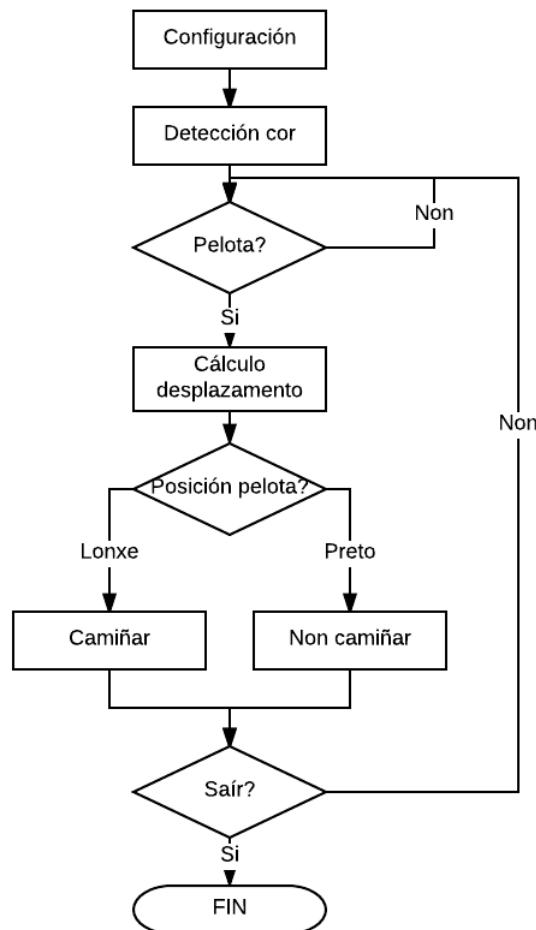


Figura 7.2.0.6 – Diagrama de fluxo para o exemplo de seguimento dun obxecto de cor

En canto á posición da cabeza, como esta no eixo X (HeadYaw) pode moverse desde -2.0857 a 2.0857 radiáns, tal e como se mostraba na táboa 6.4.1.1, considérase que a pelota está centrada cando así o está a cabeza, é dicir, cando esta última se atopa nun rango de -0.3 a 0.3 radiáns; caso no que o robot camiñará só cara a adiante. Se a cabeza se atopase virada

máis de 0.3 ou menos de -0.3 radiáns, considérase que a pelota se atopa a un dos lados e, por tanto, o robot camiñará cara a adiante e cara ao lado correspondente.

A modo de resumo, na figura 7.2.0.6 móstrase un diagrama de fluxo que recolle o funcionamento completo deste exemplo.

7.3. Seguimento dunha cara coa cabeza

Este exemplo, de forma similar ao anterior, permite ao robot Nao ser capaz de seguir unha cara coa cabeza, para o cal se volverá a facer uso das librerías de detección de imaxe e de movemento.

De forma análoga ao caso anterior, o funcionamento xeral deste exemplo pode dividirse en dúas etapas principais:

1. Detección de caras.
2. Seguimento da cara coa cabeza.

Como xa se explicou, a etapa de detección de caras realiza-se coa axuda de OpenCV e o seu recoñecedor baseado en ‘fervenzas’. Neste caso non haberá ningunha etapa de recoñecemento previo nin de calibrado, simplemente se estará agardando a que unha cara sexa detectada, e no momento en que isto ocorra, esta será seguida coa cabeza do robot.

Este seguimento consistirá en seguir o punto central da cara obxectivo, da forma mostrada na figura 7.3.0.1. Tendo en conta que no caso de que exista máis de unha cara na imaxe, a elixida para seguir será a de maior tamaño, por considerarse a más próxima ao robot.

Para o movemento da cabeza necesario para realizar o seguimento, ao igual que se facía no exemplo anterior, calcularase de forma que se manteña sempre o centro do rectángulo que contén a cara no centro da imaxe que se está obtendo en tempo real a través da cámara do robot.

Igualmente cando existan posicións de conflito nas que a cabeza poida chocar con algúns dos ombreiros, restrinxirase o movemento para evitar posibles danos no propio robot. Estas situacións problemáticas son calculadas de forma predictiva, antes de que sucedan, da forma que se mostraba na figura 7.2.0.5 e tal e como se explicou alí.

Como resumo do funcionamento deste exemplo, inclúese na figura 7.3.0.2 un diagrama de fluxo que o representa.

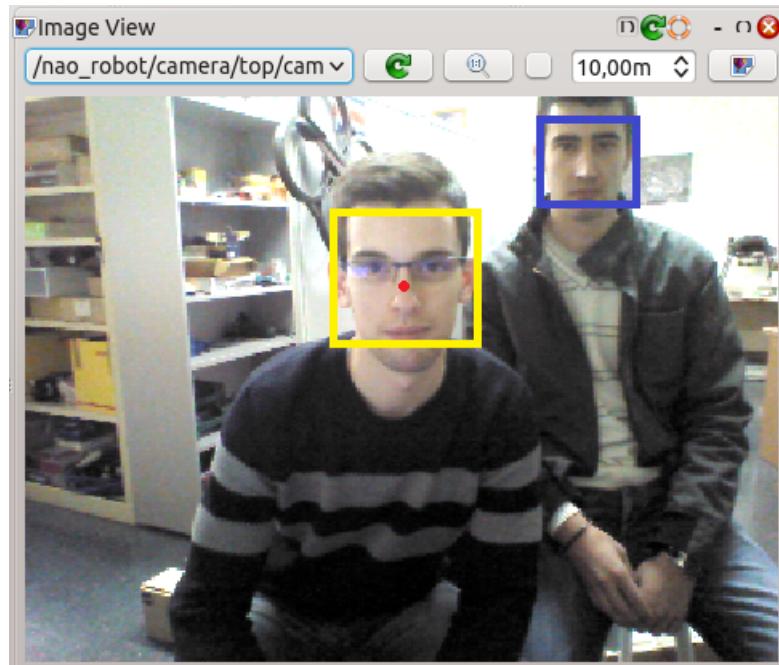


Figura 7.3.0.1 – Exemplo de detección cando se ven varias caras na imaxe

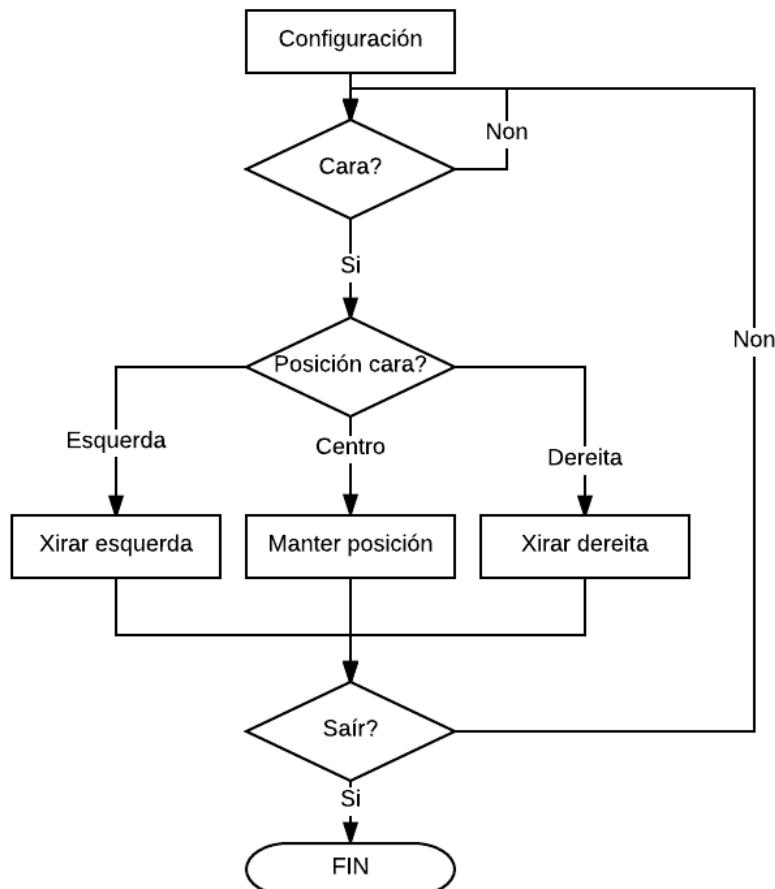


Figura 7.3.0.2 – Diagrama de fluxo para o exemplo de seguimento dunha cara coa cabeza

7.4. Seguimento e agarre dun obxecto de cor

Para a realización deste exemplo foi necesario o uso das librerías de movemento e detección de imaxe, pero facendo especial fincapé no problema de agarre de obxectos, e na precisión necesaria para facelo de forma axeitada.

As probas aquí realizadas permitirán ao robot Nao ser capaz de seguir unha pelota de cor coa cabeza e moverse cara a ela ata que a pelota se atope parada (debe de ser nun lugar e a unha altura accesibles para o robot). Desta forma, Nao será capaz de coller a pelota e andar con ela en caso de que fose necesario.

O experimento conta con unha etapa previa de recoñecemento e/ou calibrado da cor do obxecto, de forma que o rango de detección se calcule de forma automática e que dito exemplo se poida levar a cabo en condicións de luminosidade diferentes.

O funcionamento xeral pode dividirse nas seguintes etapas:

1. Detección do rango de color HSV.
2. Seguimento da pelota coa cabeza e o resto do corpo.
3. Localización da pelota no espazo 3D.
4. Agarre da pelota.

Hai que comentar que, antes de chegar ao resultado final, foron realizadas gran cantidade de probas para a verificación do proceso de agarre, nas cales se situaba a pelota a unha altura fixa e tan só se trataba de verificar ata que altura era capaz de chegar o robot para facer un agarre correcto. Comprobándose que cando esta estaba por encima da súa cabeza, o robot xa non era capaz de collela debido ás limitacións impostas pola lonxitude dos seus brazos, e porque xa non se exercía a suficiente presión sobre a pelota, motivo polo cal se lle escapaba. Por outra banda, se a pelota está demasiado baixa, o robot terá que agacharse para collela. Por estes motivos, para este exemplo estableceríonse como alturas factibles para a situación do obxecto as que se atopan entre a cintura e os ollos do propio robot.

Para o resultado final desta aplicación pártese do primeiro exemplo explicado, no cal se trataba de seguir unha pelota de cor ata atoparse a unha determinada distancia da mesma.

A etapa previa de recoñecemento da cor da pelota realizaase da mesma forma que antes, coa axuda dos botóns táctiles situados na cabeza de Nao, polo que xa non se volverá a expliar o mesmo proceso.

No que se refire á etapa de seguimento da pelota co corpo, tamén se parte do traballo realizado no outro exemplo, pero neste caso introdúcense unha serie de cambios que son explicados a continuación:

- En primeiro lugar, modifícase o valor da distancia á cal se debe para o robot respecto á pelota. Xa que neste caso, deste valor dependerá en gran parte a precisión que se teña para collela. Polo tanto, o valor elixido será de 25 centímetros, que se corresponde coa distancia á que o robot é capaz de alcanzar o obxecto se os seus brazos se atopan estirados.

Ao igual que antes, se a distancia á maior, o robot camiñará cara adiante, e no caso en que a pelota estea situada máis preto do necesario, o humanoide desprazarase cara atrás. Todo co obxectivo de situarse sempre á mesma distancia da pelota.

- Por outra parte, unha vez que o robot xa se atopa situado á distancia adecuada, este deberá orientar o seu corpo de forma que se asegure que vai ser capaz de collela, para o cal deben cumplirse dúas condicións:
 1. A cabeza e o corpo do robot deben encontrarse aliñados, de forma que o agarre se realice siempre na dirección á que apunta a cabeza e evitar así posibles erros. A posición buscada é a mostrada na figura 7.4.0.1.

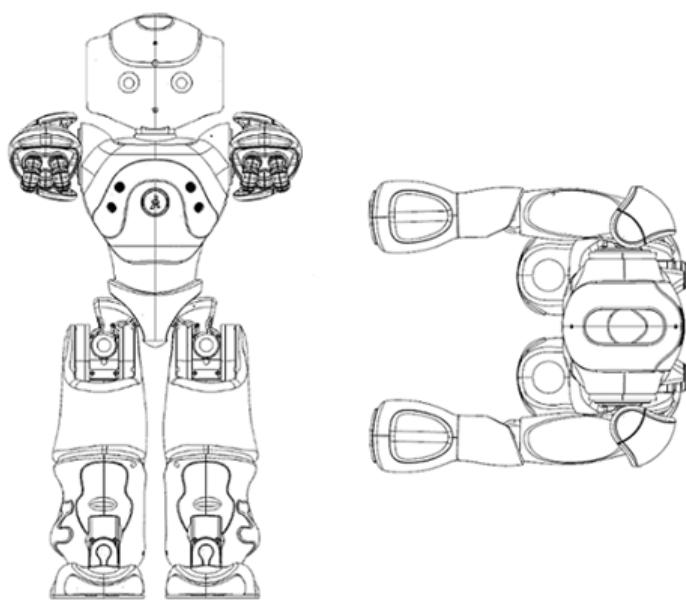


Figura 7.4.0.1 – Posición de Nao con corpo e cabeza aliñados

2. O centro da pelota debe atoparse na zona central de la imaxe vista a través da cámara superior do robot, desta forma asegurarase que o robot non choca coa pelota ao estirar os brazos para collela, xa que isto podería suceder se o robot ve a pelota nunha esquina da imaxe, tal e como se vía na figura 6.4.2.1.

Como se dixo anteriormente, o robot realizará lixeiros desprazamentos cara os lados e rotacións ata atopar a posición óptima, na cal se cumpran ambas condicións. Para isto é necesaria gran precisión, polo que opta por empregar un controlador proporcional, que lle permite ao robot realizar desprazamentos más curtos a medida que se acerca á posición obxectivo, evitándose así estar dando tombos, é dicir, que ao tratar de situarse se pase para un lado e ao ir para o outro que se volva a pasar.

A figura 7.4.0.2 mostra o código empregado para calcular estes reguladores, os cales se muestran de forma esquemática na figura 7.4.0.3, sendo o erro $e(t)$ a diferenza entre a posición central da imaxe e a coordenada X do centro da pelota, e a sinal de control $u(t)$ o producto deste erro por un valor xenérico de desprazamento calculado polo método de proba e erro. Sobra que dicir que a planta se corresponde co robot e $y(t)$ coa coordenada X do centro do obxecto.

```

if center[0]< 152 or center [0]> 168:
    e=160-center[0]
    walk.linear.y=e*(-0.005)
    walk.angular.z=-0.0

else: # Parar
    PelotaCentro=True
    walk.linear.y=0.0
    walk.angular.z=0.0
    print " Pelota no centro"

```

Figura 7.4.0.2 – Exemplo do código empregado para a implementación do regulador proporcional

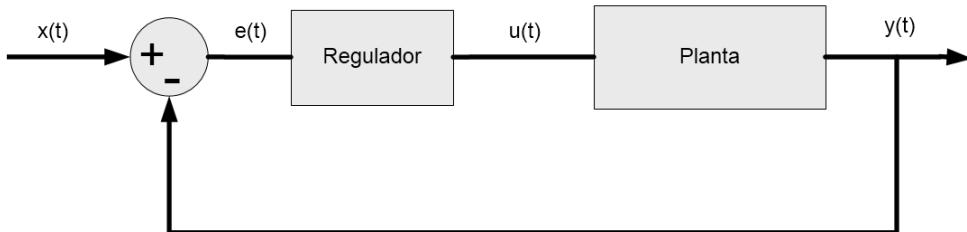


Figura 7.4.0.3 – Diagrama de bloques xenérico do regulador empleado

A continuación pasaráse á etapa de localización da pelota no espacio 3D, que como ben se explicou no anterior capítulo, consistirá en calcular as coordenadas X, Y e Z da pelota respecto ao marco de referencia situado no torso de robot, para desta forma poder calcular os movementos necesarios para o agarre en función das mesmas.

Estas coordenadas, unha vez calculadas serán publicadas como unha mensaxe tipo *Point* nun tema de ROS, de forma que se creará outro nodo que, no mesmo intre en que as coordenadas sexan publicadas, comece coa etapa de agarre. Polo tanto, neste exemplo tamén quedará clara a posibilidade de creación de dous nodos e a comunicación entre eles.

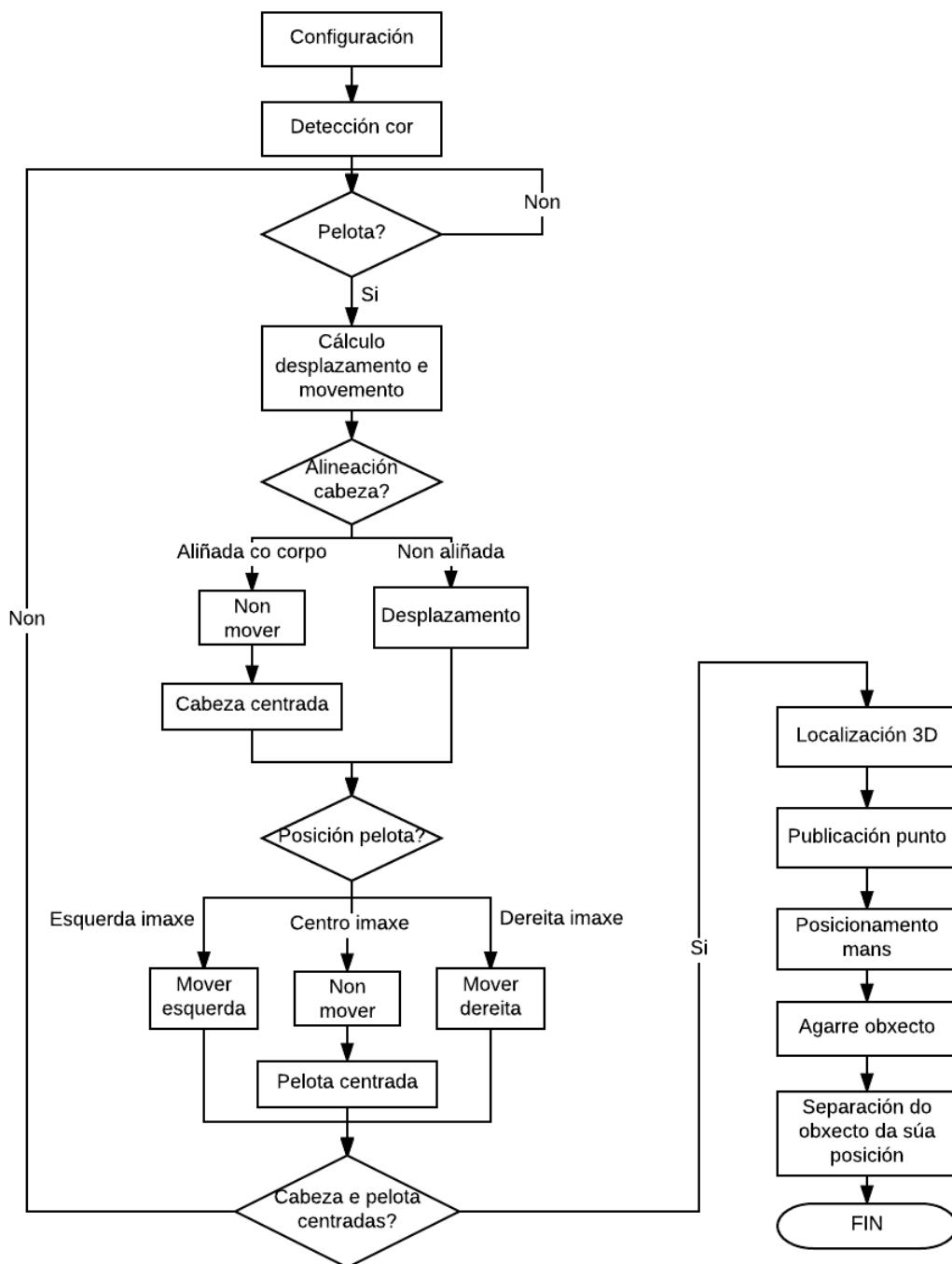


Figura 7.4.0.4 – Fluxograma para o exemplo de seguimento e agarre dun obxecto

Para finalizar, pasarase á etapa de agarre da pelota. Nesta etapa tan só se fará uso da componente Z do punto publicado, xa que pola posición final na que se situou o robot con anterioridade non é necesario facer uso das componentes X e Y. Cabe recordar que este proceso de captura se podía resumir en tres simples pasos, representados na figura 7.4.0.5:

1. Movemento de aproximación dos brazos cara a posición da pelota.
2. Agarre da pelota.

3. Separación da pelota da súa localización.

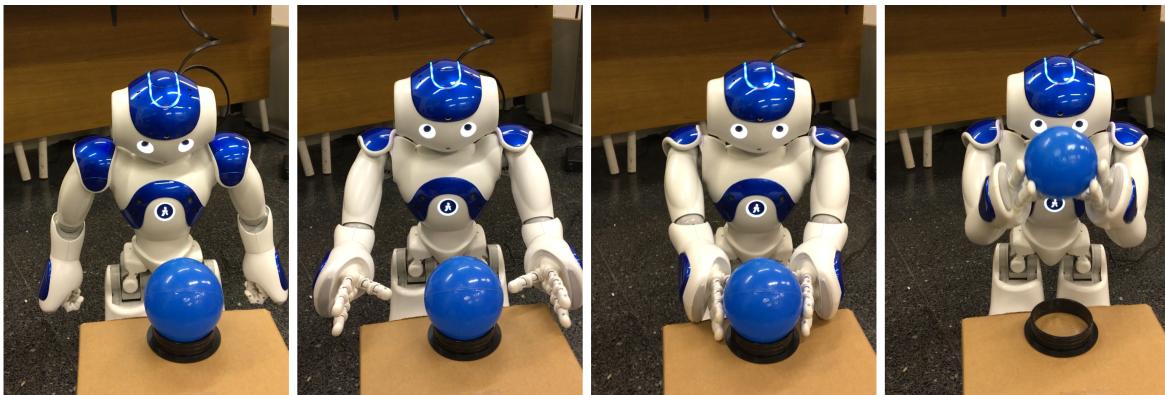


Figura 7.4.0.5 – Exemplo da secuencia de agarre de obxectos

A modo de resumo e para ver con más claridade o funcionamento deste exemplo, na figura 7.4.0.4 móstrase un diagrama de fluxo que representa todas as accións levadas a cabo.

7.5. Orientación cara a unha fonte de audio e recoñecemento da fala

Este exemplo fai uso fundamentalmente da librería de son, aplicando todas as súas funcionalidades de forma que permite ao robot Nao ser capaz de interaccionar con un humano, a través do recoñecemento de voz e da propia fala do robot, para logo ser capaz de seguir coa cabeza ou co corpo un son, unha pelota de cor vermello ou unha cara.

Tal e como nos exemplos anteriores, todo o que se facía era a través de ROS, neste caso vai a facerse a través da API de NAOqi coa que conta Nao, xa que se atopa entre os requisitos de deseño.

Por tanto, o funcionamento xeral deste exemplo pode dividirse en 2 etapas:

1. Interacción humano-robot: recoñecemento e sintetización de voz por parte do robot.
2. Seguimento dun son, pelota ou cara.

A etapa de interacción humano-robot realiza coa axuda do módulo ALSpeechRecognition e do módulo ALTextToSpeech presentes na API NAOqi Audio do robot, e cuxos principios de operación foron explicados no capítulo de ‘Desenvolvemento das solucións’.

Nesta primeira etapa o robot fará uso da súa capacidade de recoñecemento e sintetización de voz para preguntar ao usuario que é o que desexa fazer e como é que o desexa fazer. Desta forma preguntaral le se o que desexa que o robot siga é un son, unha pelota de color vermello (xa que non se inclúe etapa de recoñecemento previa) ou unha cara. A maiores tamén lle pedirá que indique se o seguimento se realizará con todo o corpo, camiñando cara o obxectivo indicado, ou se tan só será necesario o movemento da cabeza para tal fin.

Previamente á interacción realizarase a configuración dos módulos seleccionando as linguaxes para o recoñecedor e o sintetizador de voz, ademais de engadir o vocabulario a re-coñecer desexado.

Por outra parte, para facer o comportamento do robot máis parecido ao dun humano, Nao preguntará ata asegurarse de que entendeu correctamente o que ten que facer e, en caso que este non entenda o que se lle está a dicir, pedirá que se lle repita ata poder comprehendelo.

Unha vez o robot finalice a interacción e coñeza cales son os seus obxectivos configurara-se o seguidor en función dos datos proporcionados polo usuario, para logo comezar coa etapa de seguimento. Esta etapa poderá facela coa axuda da librería de son, en caso de que ese sexa o desexo do usuario, ou coa axuda da librería de detección de imaxes, se o que se ten que seguir é unha cara ou unha pelota de cor vermello.

Para esta etapa final cóntase coa axuda do módulo ALTracker presente na API NAOqi Trackers. Este módulo é un seguidor xenérico que permite ao robot seguir diferentes obxectivos (pelota, cara, punto de referencia, son) usando diferentes medios (cabeza, corpo entero, movemento, etc). Así, o obxectivo principal deste módulo é o de establecer unha ponte entre a detección de obxectivos e o movemento co fin de facer que el robot manteña a vista do obxectivo no centro da cámara, tal e como se facía nos exemplos anteriores.

O robot realizará o seguimento ata que se pare o programa (Ctrl + C).

Na figura 7.5.0.1 pode verse o fluxograma que recolle de forma global o funcionamento deste exemplo que se acaba de explicar.

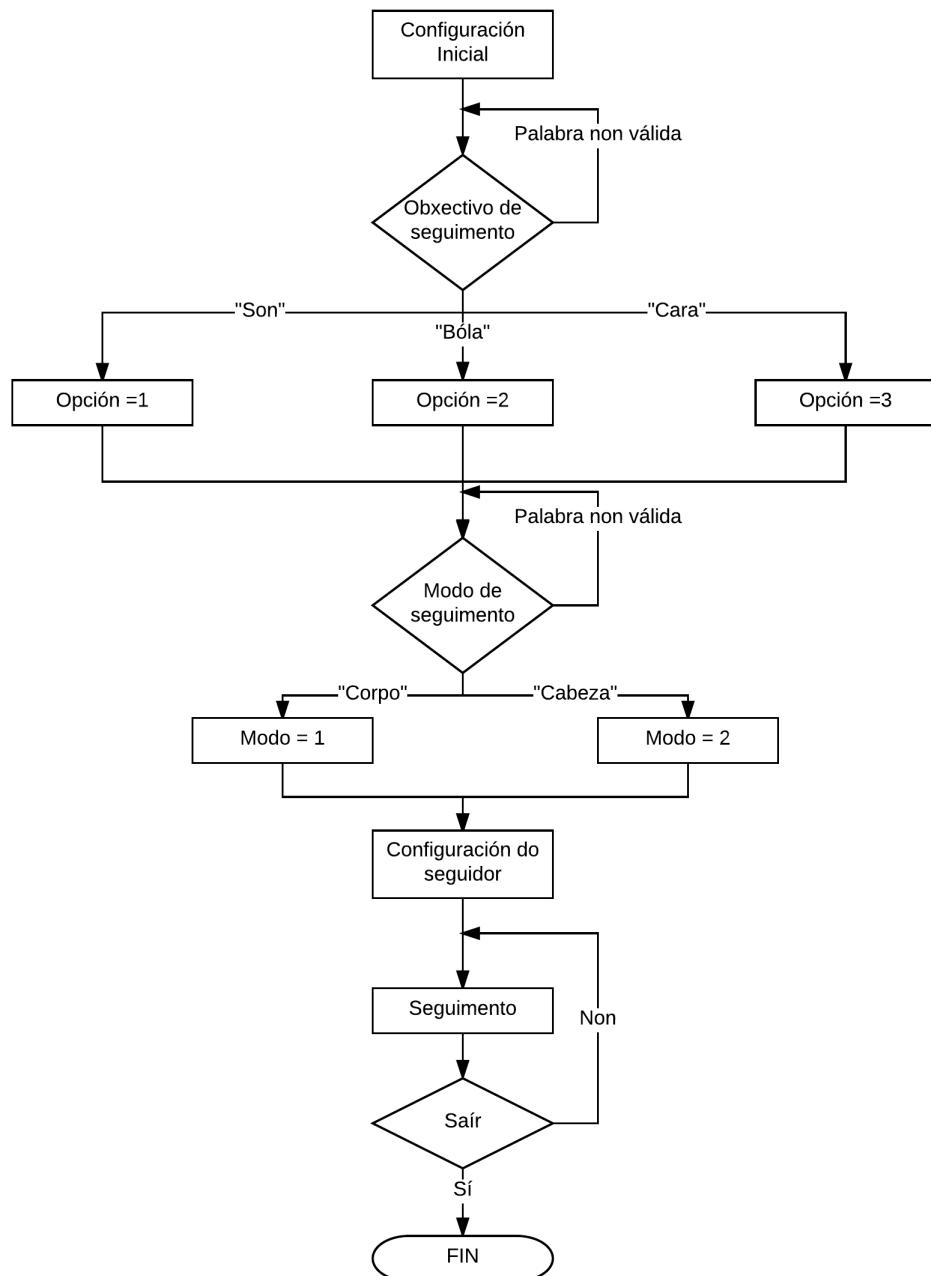


Figura 7.5.0.1 – Diagrama de fluxo para o exemplo global de son

7.6. Exemplo global de interacción con humanos

Como unha demonstración final de todo o que se fixo e todo o que se pode facer, realizaase un exemplo global de interacción con humanos, facendo para iso unha mestura de todas e cada unha das librerías de comportamento creadas, e utilizando tanto ROS como os módulos NAOqi, e todo visto a través da linguaxe de programación Python, por suposto.

Neste último exemplo desenvólvese un xogo que permite a Nao interaccionar con nenos, áinda que pode ser aplicado a calquera persoa de calquera idade. O xogo consiste nunha

combinación de todo o visto ata agora: recoñecemento de cor e obxectos, localización de fuentes de son, recoñecemento e sintetización de voz, así como todos os movementos necesarios para levar a cabo a interacción e o xogo.

En todo momento será o robot o que explique o que se vai a realizar e o que dea as instrucións necesarias ao usuario para entender o funcionamento desta aplicación. Polo tanto, traballará el mesmo facendo todo de forma autónoma.

O exemplo pode dividires nas seguintes etapas:

1. No escenario de partida, mostrado na figura 7.6.0.1, Nao estará colocado de fronte ao neno e cunha serie de pelotas de cor situadas ás súas costas. Nese momento comenzará o xogo, e pediráselle ao neno que elixa a pelota da cor que máis lle guste e que cando se atope preparado que avise a Nao. Desta forma xa entra en xogo a librería de son, a través da sintetización de voz e a través da localización da fonte de son, mentres se espera a que o usuario dea o sinal de aviso.

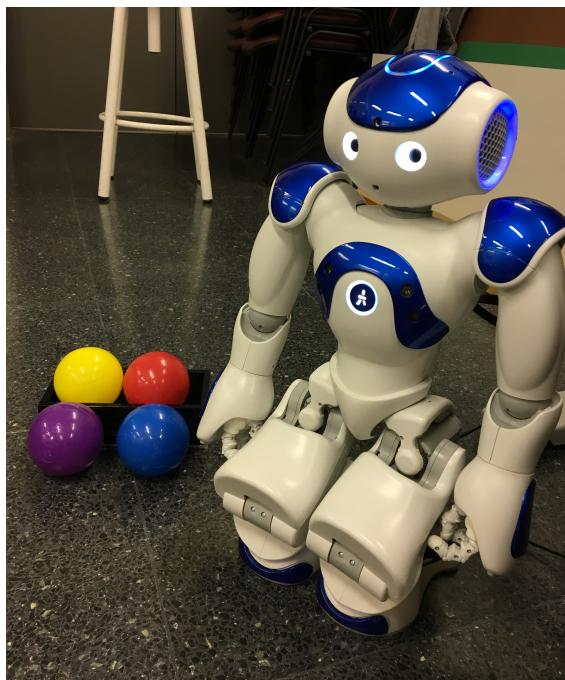


Figura 7.6.0.1 – Escenario de partido do exemplo global de interacción

No momento en que se reciba o sinal desexado, Nao virarase buscando a súa procedencia, de forma que non parará ata atopar a quen lle fala, da forma mostrada na figura 7.6.0.2.

2. Unha vez o robot teña localizado o seu obxectivo, pediralle ao neno que lle deixe ver a pelota que elixiu, de forma que colocará os seus brazos estirados para que a pelota sexa



Figura 7.6.0.2 – Nao virando a súa cabeza cara a fonte do son

depositada entre as súas mans (da forma en que se viu nos anteriores exemplos), momento no cal se procederá ao recoñecemento da súa cor, entrando en xogo a librería de detección de imaxe. Unha vez a cor sexa detectada, verificarase a través dunhas simples preguntas, e no caso de que a estimación fallase, repetirase o proceso de recoñecemento.

3. No momento en que a cor foi verificada, Nao pedirá ao mozo que se move coa pelota para que o primeiro poida seguila, e que se pare no momento en que desexe que o robot a colla, algo similar ao famoso xogo do “pilla pilla”. Empregaranse desta forma os exemplos vistos para o seguimento de obxectos e o agarre dos mesmos en canto a pelota se pare. Na figura 7.6.0.3 pode verse dende a cámara de Nao como este detecta a posición da pelota para, posteriormente, collela.
4. Finalmente, no momento no que o robot collese a pelota, como na figura 7.6.0.4, este tiraraa ao chan alegando que esa cor xa non lle gusta, e pedindo ao neno que elixa unha pelota doutra cor para volver a xogar. No caso de que o neno así o decida, repetirase de novo o xogo, mentres que se no momento en que o robot lle pregunte se desexa seguir xogando, o neno reponde que non, Nao, “moi triste”, deixará de xogar.

Nótese que en todo momento as librerías están a funcionar conviutamente, xa que continuamente o robot está a interactuar a través da librería de son, movéndose para seguir a



Figura 7.6.0.3 – Imaxe do que está a ver Nao cando se detecta a bóla

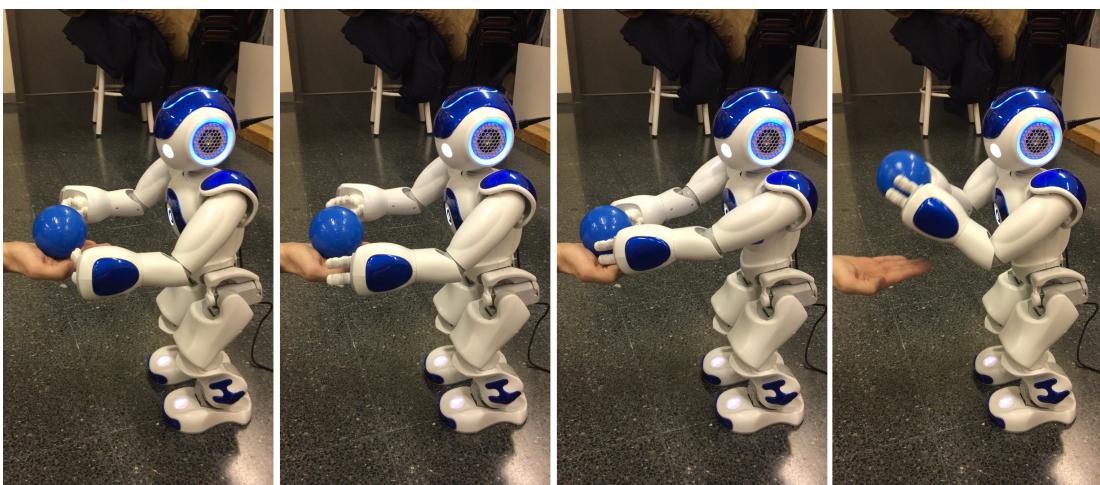


Figura 7.6.0.4 – Nao realiza o agarre da pelota que se atopa nas mans do usuario

pelota e detectando a pelota da cor desexada para poder ir tras ela.

Como resumo final deste exemplo, na figura 7.6.0.5 mostrase un diagrama de fluxo co seu funcionamento.

A maiores, é necesario comentar que este experimento foi levado a cabo con nenos de idades comprendidas entre os 5 e os 8 anos, de forma que ditas probas serviron para realizar un axuste máis detallado do código e do comportamento de Nao, ao ter en conta as particularidades da resposta dos mozos. Desta forma deuse un paso adiante no que á interacción humano-robot se refire, xa que non só foi plantexado o experimento, senón que foi probada a súa valía con persoas reais.

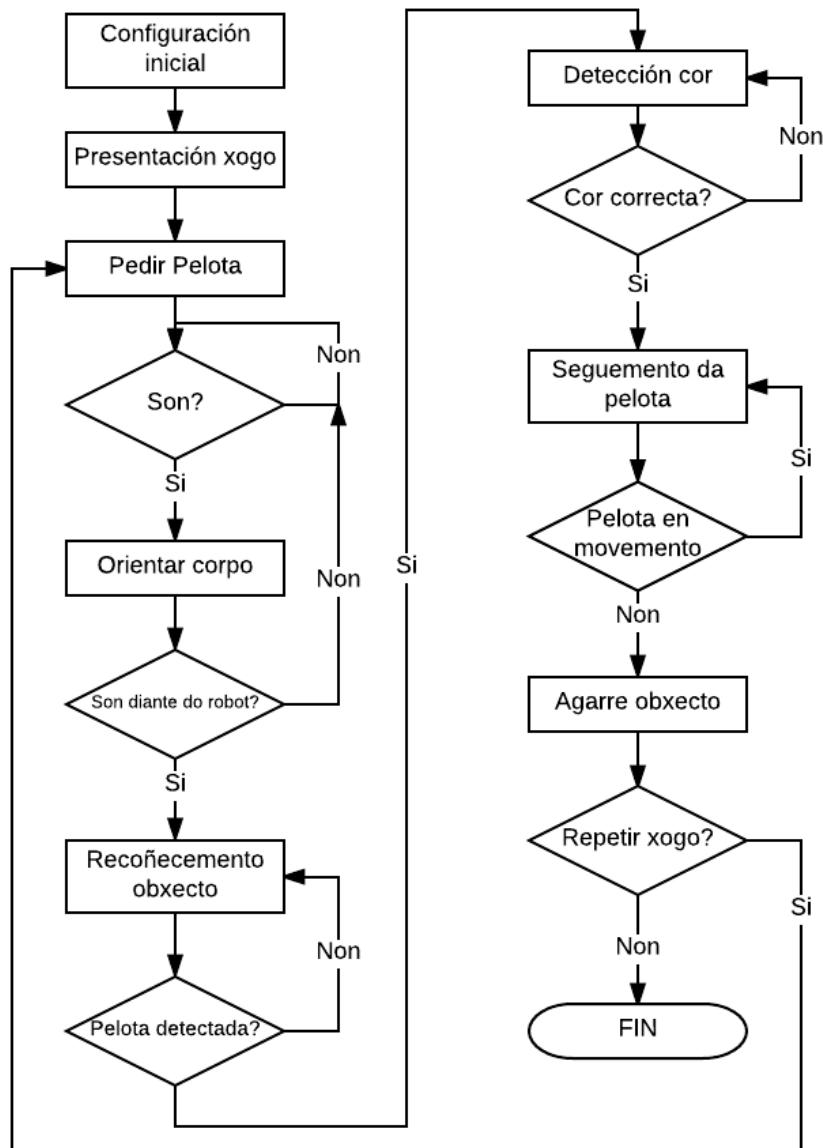


Figura 7.6.0.5 – Fluxograma do exemplo global de interacción con humanos

8 Conclusións e traballo futuro

Á vista de todas as probas realizadas, pódense sacar as seguintes conclusións acerca de cada unha das librerías de comportamentos básicos desenvolvidas e implementadas sobre o robot Nao:

- **Librería de detección de imaxe.** Deseñouse e implementouse unha librería que permite detectar obxectos de cor e caras, e que proporciona unha retroalimentación a través dunha imaxe na que se indica que é o que se está a recoñecer, de forma que o usuario poida comprobar o seu correcto funcionamento. Ambos módulos foron realizados coa axuda da librería OpenCV.
- **Librería de son.** Estudáronse diversas posibilidades para a súa realización, como por exemplo o uso da librería PocketSphinx, aínda que finalmente se optou polo uso das APIs de NAOqi, xa que, a priori, para obter uns resultados similares, o seu uso era más sinxelo. Polo tanto, implementáronse diversos módulos que permiten tanto o recoñecemento de voz como a súa sintetización, de forma que se fai posible a interacción a través da voz. Á súa vez, tamén se estudaron diferentes métodos para detectar a procedencia da fonte de son e para realizar o seu seguimento, conseguíndose ambos obxectivos.
- **Librería de movementos básicos.** Deseñáronse e implementáronse diversas funcionalidades que permiten ao robot realizar unha serie de movementos básicos, como pode ser camiñar, mover as súas diferentes articulacións ou mesmo agarrar obxectos situados a diferentes alturas. Á súa vez tamén foi creado un algoritmo que permite a estimación da posición dun obxecto no espacio 3D respecto ao marco de referencia situado no torso do robot, algo que non existía en ROS.

Para comprobar a validez das diferentes librerías desenvolvidas, así como as súas capacidades para dotar aos desenvolvedores dunha multitud de opcións de interacción humano-robot coas que crear aplicacións interactivas para o robot humanoide Nao, foron realizados diversos exemplos de aplicación específicas de cada librería e unha aplicación final para a interacción con nenos. Estas aplicacións, en forma de xogos educativos e utilidades, serven para demostrar a validez do traballo aquí feito e permiten interactuar dunha forma natural e atractiva para os usuarios:

- **Seguimento de obxectos de cor ou caras.** Con estes exemplos pretendíase deixar clara a importancia da detección de imaxe dentro do campo da interacción humano-robot. Foron realizados como exemplo da combinación da librería de movementos básicos e da de detección de imaxe, de forma que robot quedou preparado para seguir tanto unha cara como un obxecto ata quedar situado a unha distancia determinada ou, pola contra, tan só seguiu coa cabeza, de forma que se presenta como unha boa forma de interacción.
- **Agarre de obxectos a diferentes alturas.** Este exemplo fixose para amosar o potencial existente na librería de movementos básicos, e para ver como unha tarefa rutineira para un humano como pode ser o agarre dun obxecto, pode requirir moito traballo e precisión para que un robot sexa capaz de levala a cabo con éxito. Nao quedou preparado para agarrar un obxecto, apartalo do lugar no que se atopaba e ter a posibilidade de camiñar con el ou colocalo noutro lugar.
- **Exemplo de interacción a través de son.** Para deixar clara outra das vías de interacción humano-robot, foi realizado este exemplo no que se pretendía que o robot fose capaz de recabar os datos necesarios para o seu funcionamento do usuario. Desta forma, a través dos módulos existentes na librería de son, lévase a cabo unha conversación ata saber con claridade que é o que o robot ten que facer e como se debe levar a cabo (neste caso elixir entre seguir un obxecto, unha cara ou un son co corpo ou coa cabeza).
- **Exemplo global de interacción con humanos.** Este exemplo busca unha interacción natural e fluída con nenos, de forma que entren en escena todas a librerías desenvolvidas: dende o recoñecemento de obxectos de cor e sons, ou a interacción mediante voz, ata o agarre de obxectos e o seu lanzamento.

En conclusión, foron desenvolvidas unha serie de librerías que permiten unha interacción básica co robot Nao, as cales foron probadas conjuntamente nos exemplos do capítulo 7. Ademais configuráronse para seu uso os diferentes sensores táctiles dispostos na cabeza do robot e empregáronse para a realización de varios exemplos, o cal podería ser xa un primeiro paso de cara ao desenvolvemento doutro tipo de librerías de interacción.

Por último, a parte das diferentes librerías realizadas, tamén foi posible comprobar a posibilidade de funcionamento conxunto das APIs de Naoqi coas que conta Nao xunto cos paquetes de ROS, que era un dos requisitos de deseño existentes.

8.1. Traballo futuro

Dada a natureza modular das diferentes librerías desenvolvidas, existe unha gran cantidade de traballo que se pode realizar a posteriori, desde melloras nas librerías actuais ata a ampliación das mesmas ou incluso a incorporación de novas librerías que permitan outros

tipos de interacción.

A continuación presentase un listado de posibles liñas de traballo futuro:

- Existe a posibilidade de desenvolver comportamentos de máis alto nivel que permitan ao robot aprender a recoñecer determinadas figuras ou rostros, e saber con que se corresponde cada unha ao ver algo similar.
- Outra posible liña de traballo futuro consistiría nunha ampliación da librería de detección de imaxe, de forma que o robot fose capaz de recoñecer unha serie de movementos básicos realizados por unha persoa e memorizalos, para logo tratar de repetilos.
- Unha posible ampliación na librería de movemento, que permita ter a posibilidade de planear os movementos do robot, de forma que este sexa capaz de realizar accións como subir unhas escaleiras ou trepar por algúun lado.
- Ser capaz de recoñecer a outros robots para poder interactuar con eles, de forma que sexa capaz de saber se o outro robot está de fronte ou de costas, e saber cales son as súas articulacións para poder realizar tarefas xuntos.
- En canto á librería de detección facial, existen gran cantidad de posibilidades de traballo futuro, pasando pola antes mencionada aprendizaxe de rostros, para tratar de recoñecer persoas, ou polo estudo das expresións faciais para aprender a responder en función delas.
- Cabe comentar tamén que, no campo da detección de obxectos, existe a posibilidade de afondar máis no asunto e de realizar gran cantidad de melloras, que poderían dar para realizar un TFM completo.

Referencias

- [1] *DREAM project*, [Consulta 25 decembro 2016]. Disponible en: <http://www.robots thatdream.eu/>
- [2] *Nao: Aldebaran Robotics*, Softbank Robotics Europe, [Consulta 25 decembro 2016]. Disponible en: <https://www.aldebaranrobotics.com/en/cool-robots/nao>
- [3] *Robot Operating System (ROS)*, Open Source Robotics Foundation, [Consulta 25 decembro 2016]. Disponible en: <http://www.ros.org/>
- [4] *ROS wiki*, Open Source Robotics Foundation, [Consulta 25 decembro 2016]. Disponible en: <http://wiki.ros.org/>
- [5] *Python*, Python Software Foundation, [Consulta 25 decembro 2016]. Disponible en: <https://www.python.org/>
- [6] *OpenCV (Open Source Computer Vision)*, Itseez, Inc., [Consulta 25 decembro 2016]. Disponible en: <http://opencv.org/>
- [7] GOODRICH, M. A., E SCHULTZ, A. C.; *Human-Robot Interaction: A Survey*, Foundations and Trends in Human-Computer Interaction, (2007), Vol.1, No. 3, pp. 203-275.
- [8] IEEE ROBOTICS AND AUTOMATION SOCIETY. (2015). Proceedings on human-robot interaction. New York, NY: Author.
- [9] SHERIDAN, T. B.; *Human-Robot Interaction: Status and Challenges*, Human Factors: The Journal of the Human Factors and Ergonomics Society, (2016), Vol. 58, No. 4, pp. 525-532.
- [10] *Robot kills worker at Volkswagen plant in Germany*, The Guardian, [Consulta 25 decembro 2016]. Disponible en: <https://www.theguardian.com/world/2015/jul/02/robot-kills-worker-at-volkswagen-plant-in-germany>
- [11] CORRALES, J. A., GARCÍA GÓMEZ, G. J., TORRES, F., E PERDEREAU, V.; *Cooperative Tasks between Humans and Robots in Industrial Environments*, International Journal of Advanced Robotic Systems, (2012), Vol. 9, No. 94, pp. 1-10.
- [12] *Baxter*, Rethink Robotics, [Consulta 25 decembro 2016]. Disponible en: <http://www.rethinkrobotics.com/baxter/>

- [13] VERTUT, J., E COIFFET, P.; *Robot technology: Vol. 3a. Teleoperators and robotics evolution and development*, Englewood Cliffs, Ed. Prentice Hall, (1984).
- [14] SKAAR, S. B., E RUOFF, C. F.; *Progress in astronautics and aeronautics: Vol. 161. Teleoperation and robotics in space* Reston, VA: American Association of Aeronautics and Astronautics, (1994).
- [15] SHERIDAN, T. B., E VERPLANK, W. L.; *Human and computer control of undersea teleoperators* Man-Machine Systems Laboratory report, Massachusetts Institute of Technology, Cambridge, (1978).
- [16] BURKE, J. L., E MURPHY, R. R.; *The safe human-robot ratio*, Human-robot interaction in future military operations, (2010), pp. 31-49.
- [17] BARNES, M. J., E JENTSCH, F. G.; *Human-robot interaction in future military applications*, Hampshire, Ed. Ashgate. (2010).
- [18] Defense Advanced Research Projects Agency (DARPA), U.S. Department of Defense, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.darpa.mil/>
- [19] *Atlas-The Agile Anthropomorphic Robot*, Boston Dynamics, [Consulta 25 decembro 2016]. Dispoñible en: http://www.bostondynamics.com/robot_Atlas.html
- [20] *DARPA Grand Challenge*, U.S. Department of Defense, [Consulta 25 decembro 2016]. Dispoñible en: <http://archive.darpa.mil/grandchallenge/>
- [21] *World's First Self-Driving Taxis 'NuTonomy' Hit the Road in Singapore*, Techstory Media, [Consulta 25 decembro 2016]. Dispoñible en: <http://techstory.in/25082016-worlds-first-self-driving-taxis-nutonomy-hit-road-singapore/>
- [22] *Tesla driver dies in first fatal crash while using autopilot mode*, The Guardian, [Consulta 25 decembro 2016]. Dispoñible en: <https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk>
- [23] GAO, J., LEE, J. D., E ZHANG, W.; *A dynamic model of interaction between reliance on automation and cooperation in multi-operator multi-automation situations*, International Journal of Industrial Ergonomics, (2006), Vol. 36, pp. 512-526.
- [24] LEE, J. D., E SEE, K. A.; *Trust in automation: Designing for appropriate reliance*, Human Factors, (2004), Vol. 46, pp. 50-80.
- [25] SCHOETTLE, B., E SIVAK, M.; *Potential impact of self-driving vehicles on household vehicle demand and usage (Report 2015-3)*, Ann Arbor: University of Michigan Transportation Research Institute, (2015).
- [26] SEPPELT, B. D., E LEE, J. D.; *Making adaptive cruise control (ACC) limits visible*, International Journal of Human-Computer Studies, (2007), Vol. 65, pp. 183-272.

- [27] CROFT, K.; *NASA advances single pilot operations concepts: Super dispatcher fills in for a dozen first officers*, Aviation Week and Space Technology, (2015).
- [28] *Definition of Service Robots*, International Federation of Robotics (IFR), [Consulta 25 de cembro 2016]. Dispoñible en: <http://www.ifr.org/service-robots/>
- [29] *PARO Therapeutic Robot*, PARO Robots U.S., Inc., [Consulta 25 decembro 2016]. Dispoñible en: <http://www.parorobots.com/>
- [30] *Project ROMEO*, Aldebaran Softbank Group, [Consulta 25 decembro 2016]. Dispoñible en: <http://projetromeo.com/en/development>
- [31] FASOLA, F., E MATARIĆ, M. J.; *A socially assistive robot exercise coach for the elderly*, Journal of Human-Robot Interaction, (2013), Vol. 2, No. 2, pp. 3-32.
- [32] FEIL-SEIFER, D. J., E MATARIĆ, M. J.; *Ethical principles for socially assistive robotics*, IEEE Robotics & Automation Magazine, (2011), Vol. 18, No. 1, pp. 24-31.
- [33] *Flex Robotic System: Expanding the reach of surgery*, Medrobotics Corporation, [Consulta 25 decembro 2016]. Dispoñible en: <http://medrobotics.com/gateway/flex-system-int/>
- [34] *Miniaturising robotics design*, Cambridge Consultants, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.cambridgeconsultants.com/media/press-releases/miniaturising-robotics-design>
- [35] NELSON, B. J., KALIAKATSOS, I. K., E ABBOTT, J. J.; *Microrobots for Minimally Invasive Medicine*, Annual Review of Biomedical Engineering, (2010), Vol. 12, pp. 55-85.
- [36] *Sony Aibo: The history of the robotic dog*, SONY Corporation, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.sony-aibo.com/>
- [37] VLAHOS, J.; *Goodbye imaginary friends; hello A.I. dolls*, New York Times Magazine, (2015), pp. 44.
- [38] TURKLE, S.; *Evocative objects: Things we think with.*, Cambridge, Ed. MIT Press, (2011).
- [39] *Scratch language*, Scratch project, [Consulta 25 decembro 2016]. Dispoñible en: <https://scratch.mit.edu/>
- [40] *LOGO*, LOGO Foundation-MIT, [Consulta 25 decembro 2016]. Dispoñible en: <http://el.media.mit.edu/logo-foundation/>
- [41] *Robots in the classroom help autistic children learn*, BBC, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.bbc.com/news/education-20252593>
- [42] *AskNAO*, Aldebaran Robotics, [Consulta 25 decembro 2016]. Dispoñible en: <https://asknao.aldebaran.com>

- [43] *Robots found in the classroom*, Active Robots, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.active-robots.com/our-blog/robots-found-in-the-classroom/>
- [44] LÓPEZ RECIO, D., MÁRQUEZ SEGURA, E., MÁRQUEZ SEGURA, L., E WAERN, A.; *The NAO models for the elderly*, Human-Robot Interaction (HRI), ACM/IEEE International Conference, (2013), pp. 187-188.
- [45] *A first in France : a robot-coach for the Elderly*, Ville d'Issy-les-Moulineaux, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.issy.com/en/node/11560>
- [46] *Japanese bank introduces robot workers to deal with customers in branches*, The Guardian, [Consulta 25 decembro 2016]. Dispoñible en: <https://www.theguardian.com/world/2015/feb/04/japanese-bank-introduces-robot-workers-to-deal-with-customers-in-branches>
- [47] *Polite robots show glimmer of self-awareness*, Popular Science, [Consulta 25 decembro 2016]. Dispoñible en: <http://www.popsci.com/polite-robots-show-glimmer-self-awareness>
- [48] *'Autobiographical memory' lets robots act as knowledge go-betweens for ISS crews*, GIZMAG, [Consulta 25 decembro 2016]. Dispoñible en: <http://newatlas.com/autobiographical-memory-robots-knowledge-iss/39312/>
- [49] *Unveiling of NAO Evolution: a stronger robot and a more comprehensive operating system*, Aldebaran Robotics, [Consulta 25 decembro 2016]. Dispoñible en: <https://www.ald.softbankrobotics.com/en/press/press-releases/unveiling-of-nao-evolution-a-stronger-robot-and-a-more-comprehensive-operating>
- [50] VIOLA, P., E JONES, M.; *Rapid Object Detection using a Boosted Cascade of Simple Features*, Conference on Computer Vision and Patern Recognition, (2001).