Development of a general purpose tour-guide robot able to learn routes from people and to adapt and move in unstructured and crowded environments

Víctor Álvarez Santos

DEPARTAMENTO DE ELECTRÓNICA E COMPUTACIÓN

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Departamento de Electrónica e Computación

Thesis

DEVELOPMENT OF A GENERAL PURPOSE TOUR-GUIDE ROBOT ABLE TO LEARN ROUTES FROM PEOPLE AND TO ADAPT AND MOVE IN UNSTRUCTURED AND CROWDED ENVIRONMENTS

Author:

Víctor Álvarez Santos

PhD. supervisors:

Roberto Iglesias Rodríguez

Xosé M. Pardo López

March 2014

Roberto Iglesias Rodríguez, Profesor Titular de Universidade da Área de Ciencias da Computación e Intelixencia Artificial da Universidade de Santiago de Compostela e investigador adscrito ó Centro Singular de Investigación en Tecnoloxías da Información (CITIUS)

Xosé M. Pardo López, Profesor Titular de Universidade da Área de Linguaxes e Sistemas Informáticos da Universidade de Santiago de Compostela e investigador adscrito ó Centro Singular de Investigación en Tecnoloxías da Información (CITIUS)

Como directores da tese de Doutoramento titulada "**DEVELOPMENT OF A GENERAL PURPOSE TOUR-GUIDE ROBOT ABLE TO LEARN ROUTES FROM PEOPLE AND TO ADAPT AND MOVE IN UNSTRUCTURED AND CROWDED ENVIRONMENTS**" presentada por D. Víctor Álvarez Santos, alumno do Programa de Doutoramento en Investigación en Tecnoloxías da Información:

Autorizan a presentación da tese indicada, considerando que reúne os requisitos esixidos no artigo 34 do regulamento de Estudos de Doutoramento, e que como Director da mesma non incurre nas causas de abstención establecidas na lei 30/1992.

March 2014

Roberto Iglesias Rodríguez	**Xosé M. Pardo López**	**Víctor Álvarez Santos**
Director da tese	Director da tese	Autor da tese

Acknowledgment

I would like to express my gratitude to all those people who made this thesis possible, which includes the support received from my family and friends.

A very special thanks goes to my PhD. supervisors, Roberto Iglesias Rodríguez and Xose M. Pardo López, whose expertise added considerably to my graduate experience. Moreover, I would like to thank Carlos Vázquez Regueiro, whose knowledge and advices are also reflected in the results of this thesis.

I would like to thank the *Department of Electronics and Computation*, and the research centre *CITIUS* for providing me with the necessary resources for my research.

Finally, this thesis would have not been possible without the funding provided from the research projects *Intelligent and distributed control environment for a fast and easy deployment of robots in unknown environments*(TIN2009-07737) and *Robots that learn from you, as you do* (TIN2012-32262).

March 2014

Contents

Resumo

Contexto e obxectivos

Nas dúas últimas décadas os robots industriais consolidaron a súa presenza en factorías e outros ambientes profesionais, destacando as fábricas de automóbiles nas que a súa presencia foi clave para a súa competitividade. Non obstante, espérase que nos próximos anos os robots de servizo acaden un crecemento exponencial, aterrando no mercado de consumo e conquistando as nosas casas e oficinas, tal e como se enxerga en moitas obras ficción que amosan robots traballando conxuntamente coas persoas: cociñando, camiñando polas rúas ou facendo a limpeza dun apartamento. Sen embargo, para que estas predicións se fagan realidade aínda quedan moitos retos que debemos resolver, dende aqueles referidos ó hardware (autonomía, motores, capacidade de procesamento, etc), como aqueles referidos ó software. Os robots de servizo deben ser capaces de adaptarse á natureza dinámica dos ambientes domésticos. Nestes ambientes as persoas interactuan libremente e crúzanse no camiño dos robots continuamente, polo que será necesario dotar a estes robots dunhas capacidades de interacción e uns mecanismos de seguridade moito máis avanzados cos dun robot industrial. Ademáis nun ambiente doméstico o mobiliario pode cambiar de lugar con moita frecuencia e as condicións de iluminación son moi variables.

Particularmente, nos últimos anos, os robots guía convertéronse nunha área de investigación moi popular da robótica, xa que o seu deseño implica afrontar moitos dos retos presentes en calquera robot de servizo. Esencialmente, un robot guía debe ser capaz de interactuar coas persoas que poidan demandar rutas deste robot, para o cal ten que dispoñer dunha percepción fiable da contorna na que opera e ser capaz de adaptarse ás condicións cambiantes da mesma. En liña con estes retos, o obxectivo desta tese é construír un robot guía de propósito xeral coa capacidade de aprender rutas mentres segue a un instrutor. A diferenza doutros robots guía, esta capacidade permitirá obter un robot que poida ser usado en calquera evento cunha

intervención mínima dun experto en robótica, xa que calquera persoa poderá actuar como instrutor deste robot, por exemplo un membro do persoal do evento no que está operando. Este obxectivo fai necesario o desenvolvemento de comportamentos robustos ó mesmo tempo que flexibles.

Máis concretamente, nesta tese deseñamos e desenvolvemos un robot guía con comportamentos modulares e reutilizables noutros robots. Deste xeito o núcleo do robot é unha máquina de estados que activa ou desactiva os diferentes comportamentos necesarios para a gravación e reprodución de rutas do noso robot. As contribucións principais desta tese son:

- Un novidoso comportamento de seguimento de persoas que inclue un algoritmo de detección de persoas e o rastreo e a identificación do obxectivo do robot.
- Un esquema de interacción entre as persoas e o robot de comunicación bidireccional especialmente deseñado para entornos ruidosos.
- Un proceso de aprendizaxe e reprodución de rutas baseado nun novidoso algoritmo de localización multi-sensorial de gran robustez e precisión.
- Unha estratexia de aprendizaxe por reforzo que permite que o robot aprenda novos controladores adaptados a cada caso particular, máis concretamente aprendimos un controlador capaz de seguir unha persoa de forma máis intelixente que o deseñado ad-hoc.

O seguimento de persoas

O comportamento de seguimento de persoas que propoñemos consta de dúas partes clave: a detección de persoas e a distinción do instructor respecto do resto de persoas.

Por un lado, o obxectivo da detección de persoas que propoñemos é evitar que as persoas que queiran interactuar co robot deban levar calquera dispositivo ou vestimenta identificativa e ó mesmo tempo permitir o seu libre movemento diante do robot. Estes obxectivos descartan dous dos métodos máis usados ata agora para detectar persoas dende un robot: a detección mediante dispositivos RFID e a detección de caras. Polo que para acadar estes obxectivos propuxemos dous métodos diferentes: empregar un algoritmo estado da arte para a detección de persoas sobre imaxes RGB e e un detector deseñado por nós que traballa sobre imaxes de profundidade. Logo da análise das vantaxes e inconvenientes de ámbalas dúas propostas concluímos que o detector sobre imaxes de profundidade acercábase moito máis a esa solución sen restriccións que buscamos. A diferenza do detector baseado en imaxes RGB, o detector

que nós propoñemos é capaz de detectar persoas sen necesidade de que o seu corpo estea completamente visible na imaxe, como por exemplo durante oclusións parciais por parte doutras persoas ou obxectos, ou incluso cando a persoa se atopa nos bordes da imaxe. Outra vantaxe moi importante da nosa proposta é que permite facer unha segmentación persoa/non-persoa dos píxeles da imaxe moi precisa e sinxela. Por último, para mellorar a robusteza da detección de persoas implementamos un detector de pernas que usa a información proporcionada polo escáner láser do robot para buscar obxectos de tamaño similar ó dunha perna. Esto permitirá comparar a posición das posibles pernas detectadas no escáner láser cos corpos detectados na imaxe de profundidade, que no caso de coincidir aumentaríamos a confianza da detección realizada. Tanto o detector de persoas en imáxes de profundidade como o detector de pernas están baseados na búsqueda de diferenzas de profundidades, é dicir en ámbolos dous casos escaneamos os datos buscando diferenzas relevantes entre puntos adxacentes. Estas diferenzas indicannos os bordes dos obxectos, que despois soamente deben ser filtrados polas súas características de forma ou tamaño. Ámbolos dous métodos ofrecen unha alta taxa de recoñecemento.

Por outro lado, o obxectivo do proceso de distinción do instructor que propoñemos é evitar que mentres o robot se atopa seguindo a un instructor, para a aprendizaxe dunha nova ruta, non se confunda con outra persoa que se poda cruzar no seu camiño e comeze a seguir a esta outra persoa. A distinción de personas que propoñemos está baseada na súa posición pero sobre todo na comparación da cor e textura da súa vestimenta. Para poder realizar esta comparación é necesario un descritor visual robusto e adaptable ás condicións cambiantes dun ambiente doméstico. Para acadar este obxectivo, seguimos os seguintes pasos:

1. Nun primeiro paso, construímos un amplo conxunto de características visuais (tanto de cor como de textura) que consideramos adecuadas para empregar. Compre recordar as restricción de tempo de procesamento existentes nun robot coma este, polo que por unha parte descartamos incluír neste primeiro conxunto inicial algunhas características coas que dificilmente se pode traballar en tempo real, e por outra parte tampouco podemos empregar tódalas características deste primeiro conxunto.

2. Nun segundo paso, reducimos o conxunto de características inicial ata acadar un grupo de 8 características que proporcionan a maior cantidade de información coa mínima redundancia posible. Para coñecer a redundancia das características fixemos unha análise completa baseada na información mutua que comparte cada característica co resto. Esta análise permitiunos crear un ranking das características en función da súa redundancia

coas anteriores e así establecer subconxuntos cos que medimos o seu rendemendo no noso problema. O resultado desta análise foi que un conxunto de 8 características ofrecía o rendemento máis alto ó mesmo tempo que as súas características apenas contiñan información redudante. O conxunto resultante está composto polo espazo de cor HSV, as segundas derivadas da imaxe en ámbolos dous sentidos (vertical e horizontal), un descritor da densidade de bordes (ED), o descritor csLBP e un histograma de bordes baseado no do MPEG-7.

3. Por último propuxemos un algoritmo que é capaz de variar dinamicamente a relevancia de cada unha das características deste conxunto, de forma que en cada situación se empregue o subconxunto máis adecuado de características visuais. Para poder variar a relevancia de cada característica do conxunto é necesario poder medir a súa eficacia en cada momento, polo que propuxemos un método que consiste na comparación da distribución da característica nos dous individuos. Non obstante para realizar esta comparación existen diferentes métodos polo que levamos a cabo unha análise de todos eles para concluír que a nosa proposta se atopa entre as mellores.

Para establecer unha comparación entre os descritores dunha persoa e os de outra computamos os histogramas de cada unha das características que compón o noso descritor e comparámolas usando a coñecida distancia de Bhattacharyya.

Finalmente, unha vez que o robot conta coa capacidade de detectar persoas e identificar e rastrear o seu instructor soamente restaba crear un controlador que permitise ó robot moverse cara ó obxectivo (o instructor) ó mesmo tempo que esquiva os obstáculos cos que se atopa. Para isto creamos un controlador reactivo baseado en campo de potenciais, no que a persoa exerce de forza atractiva e os obstáculos de forza repulsiva.

Interacción persoa-robot

O obxectivo da interacción entre a persoa e o robot é, de novo, que sexa útil no maior número de ambientes posibles, polo que debe ser unha solución non dependente das características do ambiente no que o robot está operando. Por esta razón descartamos o recoñecemento de voz que é moi dependente do ruído, e por tanto pouco útil en lugares como eventos e exposicións, lugar de traballo habitual do robot-guía que pretendemos construír. O deseño desta interacción está baseado en varias probas de usuario documentadas nesta tese (que seguen o estándar

ISO/IEC-9126 para avaliar a calidade do software), así como na nosa experiencia observando ós visitantes dos eventos nos que participamos co robot.

Podemos dividir o esquema de interacción entre as persoas e o robot que deseñamos segundo a dirección da información:

– A comunicación do robot á persoa: realizada mediante unha interface gráfica de realidade aumentada (AR-GUI, polas súas siglas do inglés) e *feedback* mediante mensaxes de voz pre-gravados. A AR-GUI permite que os usuarios do robot podan verse a si mesmos, ó mesmo tempo que observan o que percibe o robot en cada momento, así como coñecer o que espera o robot de cada usuario (coa axuda do feedback de voz). Este elemento é moi importante xa que permite que calquera persoa poida entender de xeito sinxelo o que está ocorrendo e desta forma axudar ó robot nas suas tarefas. Un exemplo moi claro é que se o robot deixa de seguirnos podemos comprobar cal foi o problema dun vistazo rápido. As probas levadas a cabo en eventos de demostración do robot evidenciaron que a gran maioría dos usuarios dun robot están dispostos a axudar á un robot sempre e cando sexan conscientes do que está a acontecer.

– A comunicación da persoa ó robot: realizada mediante xestos (movementos coas máns) en fronte do robot. Deseñamos e implementamos un detector e un *tracker* de mans axudado pola AR-GUI. Para que o robot detecte as mans da persoa, esta deberá erguelas cara o robot cando se atope en fronte del. Se o robot detecta unha man debuxará unha bola de cor en tres dimensións sobre a súa posición. Esta bola permitirá ó usuario do robot coñecer se o robot está a seguir as súas mans, e polo tanto se pode realizar xestos. A nosa proposta para a detección de xestos é moi sinxela para non sobrecargar o procesamento do robot: empregamos unha máquina de estados baseada na posición da man para cada un dos xestos que recoñecemos: movemento horizontal de *adeus*, movementos de *swipe* tanto a esquerda como á dereita e movemento de empuxar un botón virtual no aire.

Podemos describir a interacción típica do noso robot dende o punto de vista dun visitante ou dun instructor. No caso dun visitante, calquera persoa poderá demandar rutas para a súa demostración mediante unha sinxela operación: poñerase diante do robot ata que este detecte as súas mans e logo presionará no botón virtual de reproducir ruta (mediante un xesto coa man) e seleccionará a ruta desexada. No caso dun instructor, este indicaralle mediante un xesto coas dúas mans que o robot debe seguilo ata o punto de inicio dunha nova ruta. Neste

punto presionará un botón virtual (mediante un xesto coa man) para indicarlle ó robot que debe comezar o proceso de gravación de rutas, co cal o robot comezará a seguir a esta persoa que se converterá no instructor. O instructor poderá moverse libremente ó longo da ruta que desexa que o robot grave e incluso pararse nun punto de interese e gravar un mensaxe de voz que indicará mediante outro botón virtual. Esta mensaxe de voz será reproducida polo robot no mesmo punto que foi gravada cando a ruta sexa demandada por un visitante. Por último cando unha persoa desexe que o robot finalice o proceso de gravación ou seguimento da persoa, so terá que facer un xesto de adeus coa súa man. Este proceso de interacción é completamente guiado por parte da interface gráfica do robot e as probas que fixemos en demonstracións do robot indicáronnos que é ademais moi sinxelo e de fácil aprendizaxe por parte dos usuarios do robot.

Aprendizaxe e reproducción de rutas

O obxectivo é deseñar os procesos de gravación e reproducción do robot-guía, é dicir que información se debe gravar mentres o robot segue a un instructor e como usaremos esa información para reproducir unha ruta cando esta sea demandada por un visitante.

A clave destos procesos atópase nun novo algoritmo de localización multi-sensorial deseñado por nós. Usando un filtro de partículas, este algoritmo é capaz de fusionar información procedente de varios sensores asíncronos que envían información a diferentes taxa. Estes sensores son: un sistema de posicionamento Wi-Fi, un escáner láser, e un compás magnético. O sistema de posicionamento Wi-Fi consiste nun sistema capaz de predecir o punto no que se atopa o robot a partires das potencias de diferentes puntos de acceso Wi-Fi. A precisión deste sistema atópase xeralmente por encima dos 5 metros, polo que aínda que non contribue a localizar o robot con precisión nun ambiente doméstico, si que permite discernir con facilidade entre diferentes habitacións que, por outro lado, poden ser demasiado similares para un que un escáner láser sexa capaz de distinguilas. Polo tanto, podemos dicir que estes sensores cometen erros non correlacionados e, polo tanto, cada sensor poder suplir os inconvenientes dos outros. Este punto incrementa notablemente a robustez do sistema, actuando incluso como prevención ante fallos temporais en calquera dos sensores, como por exemplo unha alteración do campo magnético medido polo compás. Nesta tese atópase unha descripción completa do sistema de posicionamento multi-sensorial que deseñamos para o noso robot.

A robustez deste algoritmo permítenos gravar rutas de xeito moi sinxelo: simplemente temos que gardar as coordenadas polas que se move o robot xunto coa sua orientación. Esta

información será suficiente para logo reproducir as rutas por medio dun planificador. Este planificador será o encargado de dirixir o robot dende o punto no que se atopa ata o punto de inicio da ruta, os cales poden estar separados varias decenas de metros. Unha vez que o robot se atopa no punto inicial un planificador local será o encargado de mover o robot ó longo da ruta, ó mesmo tempo que se reproducirán as mensaxes de voz gravadas durante a aprendizaxe da ruta.

Aprendizaxe por reforzo

O obxectivo desta parte da tese é demostrar que calquera persoa pode ensinar a un robot partes do seu controlador mediante técnicas de aprendizaxe por reforzo. Para lograr este obxectivo, planteamos aprender un novo controlador para substituir o que deseñamos para seguir ó instructor. O controlador antigo era totalmente reactivo: se detectaba ó instructor móvíase cara el e senón quedabáse á espera. Esta aproximación pode mellorarse facendo o controlador máis intelixente e complexo, por exemplo, cada vez que o instructor se perda por un determinado ángulo o robot debería mostrar un comportamento mais proactivo buscando ó instructor nese ángulo. Mellorar un controlador manualmente para ampliar as súas capacidades pode ser un proceso complexo, requerindo un experto en robótica.

A solución que propoñemos nesta tese consiste nunha aprendizaxe por reforzo mediante un joystick. Calquera persoa poderá indicarlle ó robot cando está realizando unha acción incorrecta ata que o robot aprenda cales son as accións que os seus usuarios esperan del en cada momento. Nesta tese descríbese o algoritmo de aprendizaxe que propoñemos, así como os resultados que obtivemos durante as probas que realizamos no laboratorio do CITIUS. Estes resultados demostraron que en apenas 5-10 minutos o robot é capaz de aprender a seguir unha persoa, esquivar obstáculos e ademais buscar a persoa no mesmo ángulo no que a perdeu.

A aprendizaxe deste ou doutros controladores permiten que calquera persoa poida dotar ó seu robot de comportamentos moito máis avanzados dos que conta nun inicio, ademais de adaptalos á contorna na que traballa a diario. Un robot que conte coa capacidade de aprendizaxe será capaz de reaccionar correctamente ante situacións non contempladas nun inicio, co que cumpliremos co obxectivo de construír un robot menos dependente do entorno no que traballa.

Demonstracións realizadas

Dado que o obxectivo desta tese consistía en dar un paso adiante nos robots de servizo, sacándoos dos laboratorios e acercándoos as persoas, dende un principio procuramos realizar tódolos eventos posibles nos que mostrar as capacidades do noso robot ó mesmo tempo que nos permitía probar as súas capacidades e observar a particularidades da interacción de cada persoa co noso robot.

Deste xeito, levamos a cabo numerosos experimentos, probas e demostracións en ambientes académicos onde implementamos os comportamentos descritos (Departamento de Electrónica e Computación, Escola Técnica Superior de Enxenaría, Centro de investigación en Tecnoloxías da Información da USC). Non obstante tamén fixemos varias visitas ó museo Domus (A Coruña), onde organizamos un evento con límite de prazas que cubriu a súa inscripción en apenas un día. Visitamos moitas escolas de secundaria e primaria (en Lalín e en Santiago, dentro dun programa de divulgación científica de tres mses) e en eventos como o *día da ciencia en galego*, no que participamos durante 3 anos consecutivos. Nestas demostracións permitimos ós visitantes probar cada unha das partes do noso robot, dende o seguimento de persoas, a interacción, ata a gravación e reprodución de rutas. Estimamos que en total máis de 1000 persoas estiveron nalgunha demostración ou evento no que participamos, das cales unha gran porcentaxe puido probar o robot persoalmente.

CHAPTER 1

INTRODUCTION

More and more, robots are moving outside industry and research centers, their traditional homes. In this regard, the latest years has seen how new trends are arising in the robotics market, the rise of service and personal robots.

In this chapter we will contextualise this situation, and then, we will describe the challenges of those types of robots. This will let us introduce a specific service robot that is of great interest: the tour-guide robots. As we will describe, tour-guide robots has been an interesting problem in robotics over the latest years, therefore we will review those that have been presented in recent years, and then we will describe our goals, and its connections with the challenges of today's personal robots.

1.1 Robots at the shop-window

The population of robots is increasing every day, in number and diversity of deployment environments, at rates unthinkable just few years ago.

Until the last decade, robots could only be found at very specific industrial sectors such as the automotive. Those are classified as industrial robots. The ISO 8373:1994 standard defines an industrial robot as *an automatically controlled, re-programmable, multi-purpose manipulator programmable in three or more axes*. An industrial robot can be either mobile or fixed, although the most common ones are fixed robotic arms used for tasks such as welding, painting, assembly or packaging. Figure 1.1(a) shows an example of an industrial robot used to weld.

(a) Industrial robot (b) Professional service robot (c) Personal service robot

Figure 1.1: Examples of the different types of robots: industrial, a welding robot from ABB; professional, a pick and place agricultural robot from Harvest Automation; and personal robots, a vacuum cleaner Roomba from iRobot.

In the last decade, the advancement of technology brought us the service robots. The main characteristic of service robots is that they are able to work closely with humans, cooperating with them in their daily life at home as well as at work. The definition of a service robot has not been standardised yet, but since 2007 there is a working group of ISO to include an official definition of service robots. In the meanwhile, the International Federation of Robotics (IFR) has proposed a tentative definition: *a service robot is a robot which operates semi or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations*. Service robots can be divided into two sub-classes: professional and personal robots. Professional robots are mainly present in the agriculture (Fig. 1.1(b)), defense and passenger transport services, while personal robots are those which help humans in their homes or offices. Examples of personal robots include the robotic vacuum cleaners (Fig. 1.1(c)), robotic mowers, pool cleaning robots or tour-guide robots.

In terms of market size, service robots currently represent a tenth of the sales of industrial robots. Nevertheless, it is expected that in the next years sales of service robots will growth exponentially. The International Federation of Robotics (IFR) forecasts [1] that the global market size will increase from 3,3B US$ in 2008 to 100B US$ in 2020, which can be translated to a growth of over 3000% in those twelve years. Similarly, the Japanese Ministry of Economy, Trade and Industry and the Industrial Technology Development Organisation has also forecasted a growth of over 1700% of service robots in Japan: from €0.45B to €7,8B in the period 2011-2020 (Fig. 1.2). According to the predictions shown in Fig. 1.2, service robots sales are expected to surpass the industrial robots around year 2020.

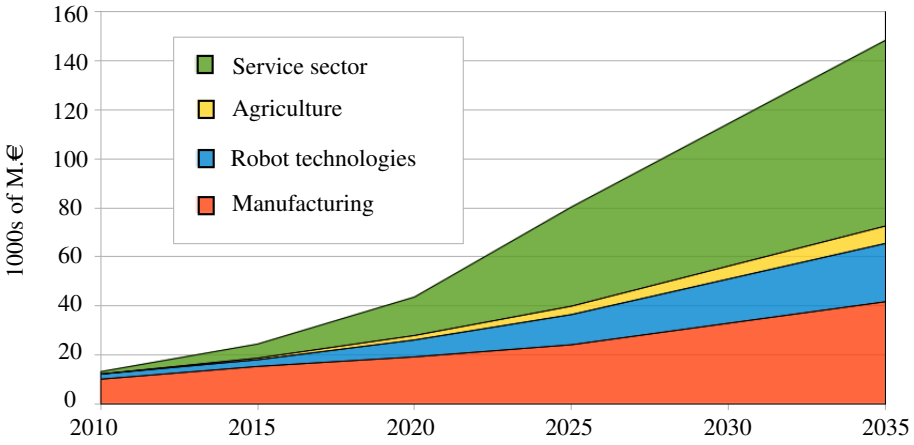


Figure 1.2: Market size estimates for the robotics sector in Japan. Source: the Japanese Ministry of Economy, Trade and Industry and New Energy and the Industrial Technology Development Organisation (NEDO).

It seems obvious that if those predictions come true, service robots will become a common element at work but also in most homes or offices, playing an important role as appliances, servants and assistances; they will be helpers in repetitive or dangerous tasks, and they will be able to act as eldercare companions. Nevertheless, there are still a number of challenges that should be overcome in order to allow these robots to reach the mass consumer market. More specifically, the robotics community [1] have identified the following ones:

C1. The development of systems that can perceive the environment and the humans in a more robust and reliable level.

C2. The development of autonomous systems that are capable of learning from the user and the robot's experience.

C3. The development of systems that are able to operate in dynamic and non structured environments.

C4. The development of interfaces that allow and ease the human-robot interaction.

C5. The development of systems that are capable of autonomously managing the robot's energy, exploring new energy sources.

C6. The development of modular architectures for the development and reuse of advances systems.

A good example of a service robot which tackles those challenges is a tour-guide robot. That is, a service robot whose purpose is to offer guided routes to visitors of a museum, exhibition or any other event. This kind of robot need to possess a high level of human-robot interaction, as well as a good estimation of his position within the environment where it operates.

In this regard, over the last fifteen years various research institutions and even companies have presented their prototypes of a tour-guide robot. In the next section we will review the most notable ones.

1.2 Tour-guide robots in the past

The first well-known guide robots were Rhino [2, 3, 4] first presented in 1995, and Minerva [5] presented in 1999. The first one, was focused on its navigation abilities with predefined routes, and a touch screen with four buttons to select a route. The second one improved the localisation and navigation but it also started exploring how the robot could attract the visitor's attention by introducing a head to express emotions. Both robots had voice feedback to communicate certain states of the robot to the nearby people, and a web interface to command the robot. Probably the main drawback of those two first tour-guide robots was its limited human-robot interaction, which required a human operator that would translate the human desires into robot's tasks. In 2003, RoboX [6], performed the first big experiment with tour-guide robots in public spaces: eleven robots accumulated 13.313 hours of operational time at Swiss National Exhibition in 2002. The robot's main characteristics include fully autonomous navigation with minimal supervision and speech interaction. Nevertheless, it is important to mention that one of the major drawbacks of RoboX was again its interaction abilities which were highly limited by the level of noise due to the large number of visitors. In 2004, the robot Jinny [7] presented an interesting improvement over the previous tour-guide robots: it's routes were semi-programmable using a web interface. In the same year, Robovie [8] was presented as, probably, the first tour-guide robot with an humanoid shape since it included two robotic arms, but it required many external sensors to navigate, interact with people and guide them throught the environment.

Given that most previous tour-guide robots failed to offer a successful human-robot interaction, researchers started to focus on this topic. In 2005, the robot Alpha [9] was presented as

a platform for human-robot interaction. It consisted on a fully humanoid robot that was standing still, pointing towards destinations according to what visitors demanded. In 2007, a museum tour-guide robot [10] was presented to study non-verbal human-robot communication. In the same year, Rackham [11] described improvements in human detection and tracking as well as a novelty in human-robot interaction: hand tracking and hand symbol recognition. In 2008, Urbano [12] was presented as a tour-guide robot with two arms and a expressive face. It could showcase routes to visitors by selecting them from a internal database. In 2009, an innovative perception system for robot localisation was introduced with CiceRobot [13], it consisted on simulating the environment that the robot would see from its position and on comparing that simulation with the its actual view. In the same year Robovie [14] was updated with improvements in its social abilities.

The tour-guide robots that we have mentioned are a few of the most relevant that have been created in research centers and universities, Nevertheless there are some companies which have also created their own prototypes. Sadly, those companies does not release much information to the public domain about their robots and only a few are known by the community. As an example, Gilberto [15] is a wheeled tour-guide robot from Bluebotics which has a touch screen for interaction. Also, the impressive REEM humanoid robots [16] from PAL Robotics are examples of service robots which can also act as guide robots. REEM robots demonstrated their skills in shopping malls at Abu Dhabi and Barcelona, as well as in several international events and conferences.

As a summary from this review, we can conclude that even tough many institutions have presented their tour-guide robots in the past, it is still not a closed problem. Human-robot interaction capabilities, including human detection, tracking and identification specially at crowded and unstructured environments is still a major challenge of this kind of service robots.

1.3 Goals and structure of this thesis

The main goal of this thesis is to develop a general purpose tour-guide robot that is able to operate in different environments, such as museums, large buildings, events or conferences, with a short deployment time.

Deployment time was not a major concern in previous tour-guide robots, this leads to the fact that route learning and environment adaptation usually took weeks of work by experts in robotics. The tour-guide robot that we are presenting will learn routes while following

Figure 1.3: The tour-guide robot following an instructor in a narrow test-environment.

an instructor (Fig. 1.3). This person could be anyone from the staff of the event where it operates, which also represents a novelty in the concept of a tour-guide robot. Later on, the robot will have to show these routes to event attendees. Therefore, the robot behaviour should be robust and flexible in order to be able to cope with the different humans, environments and situations that it might encounter. The goal of *building a general purpose tour-guide robot* can be broken down into smaller milestones and associate each one with the challenges faced by today's personal robots:

G1. Detecting, tracking and following humans efficiently in the surroundings of the robot. Chapter 2 describes the person following behaviour that takes place in this tour-guide robot. It describes how humans are detected, identified and tracked over time in order to avoid confusion between the instructor and other people (Challenge C1).

G2. Interacting with humans. This is a complex goal which addresses challenges C1 and C4: the robot will need to recognise the human's commands and it should behave according to them. Moreover, given the general purpose of the robot, this interaction should be focused on reducing the learning curve needed to operate with the robot. Chapter 3 describes both how a human can send commands to the our tour-guide robot and how does the human perceive what the robot is sensing.

G3. Recording routes that an instructor might teach to the robot in the event where it operates. This goal takes on challenges C1 and C2: the robot will need to understand the

route that the instructor is teaching and store it for future reproduction. Chapter 4 deals with this problem which mainly is a robot localisation problem.

G4. Recording voice messages at points of interest within the route. This goal completes G4 and addresses challenge C4: the instructor will need to be able to interact with the robot when he is teaching a route, in order to inform about points of interest. Chapter 4 describes this process.

G5. Reproducing routes and voice messages under command from a person. This goal tackles challenges C1,C3 and C4: first, the robot will need to understand the user's desires to select a route to reproduce. Then, it will be critical for the robot to be correctly localised in the environment that it is operating on, otherwise, the robot will not be able to reproduce the route. Finally, it will need to be able to dynamically compute paths and navigate in an environment that might be crowded or with changes or rearrangements of furniture. Chapter 5 details how we have solved this problem in the tour-guide robot.

G6. Development of a strategy to let anyone teach the robot how to behave in certain situations. In particular, it would be desirable that a controller like the one that is necessary to move the robot in order to follow a human could be teached to the robot (Challenge C2).

Moreover, developing a general purpose tour-guide robot involves that several pieces of software should work together. Therefore challenge C6 of today's service robots is also addressed by developing for a modular robot architecture like the popular ROS [17]. Precisely, the fact that five out of the six challenges of today's service robots are addressed, makes tour-guide robots a popular practical goal in robotics research. Finally, chapter 7 will present a global discussion of the results of this thesis and a description of the lines that are still open after this work.

CHAPTER 2

THE PERSON FOLLOWING BEHAVIOUR

The ability of following a person is a key element for most service robots that aim to live amongst humans. People might require the help of a robot in different locations, and probably the most natural way is to simply ask the robot to follow them to that new location. In this regard, it is very important that the robot does not mistake the person that it is following (*the target*, from now on), with any other present in the environment (which we call *the distractors*). This kind of mistakes significantly lower the perception of intelligence that an user of the robot might have about it.

In the case of our general purpose tour-guide robot, a robust person following behaviour also plays a very important role. Traditionally, route teaching to this type of robots has been a programming task performed by robotic experts. However, in our robot any instructor should be able to teach new routes by simply asking the robot to follow him. In this regard, we have worked hard to reduce mistakes between the robot's instructor (the target) and the distractors [18, 19], because those mistakes could lead to recordings that should be discarded and taught again. As a result of this behaviour, our robot will be able to follow humans in real environments, avoiding the use of wearable gadgets, special clothes or modified environments.

In this chapter, we start introducing the reader to the the challenges inherent in a person-following behaviour, and the state of the art on each of them. Then, we present the main elements of the person-following behaviour that we have designed for our tour-guide robot. Next, we will go into more detail and explain how our tour-guide robot detects humans, recognises its instructor and follows him, including all the experiments that have been done to support

our proposal. Finally, a discussion on the results that we have obtained, as well as possible future improvements will close the chapter.

2.1 Challenges

Person-following in mobile robots contains many challenges of today's service robots, and each one can be considered a research topic itself:

- The detection of a human from a mobile platform.
- The recognition and tracking of a specific human, and therefore its discrimination from the rest of the humans in the surroundings of the robot.
- The navigation of the robot towards the target while avoiding colliding with the environment.

First, in order to follow a human a robot must be able to detect those who are in its surroundings. However, detecting persons in images is not trivial: it is considered a particularly difficult object recognition problem, because people are non-rigid objects with a large number of textures and colours due to the different clothes that they might wear. Different solutions for the problem have been presented but they only apply for certain sensors and under certain conditions.

Second, a person following robot should be able to establish a robust and flexible tracking of the detected humans. It is specially important to achieve a robust human recognition behaviour that is able to distinguish between the target and the distractors. In this regard, this problem is usually faced using the position of the human, and also the colour of his clothes. Nevertheless, it can still be hard to distinguish humans when they move close to each other in real environments, which usually contain challenging illumination conditions.

Finally, the relative position of the target from the robot should be estimated. The robot should safely move towards its target, maintaining a safe distance and avoiding obstacles, as well as, other humans.

2.2 State of the art

In the last fifteen years human detection and tracking from mobile robots obtained a high relevance amongst the robotic community alongside the increasing importance of service robots.

In this regard, in 1999, one of the first robots [20] in detecting and tracking people combined different cues: colour, motion and shape analysis. These cues were extracted from images with depth information which were obtained using a stereo pair. A year later, a robot [21] was able to detect humans using motion detection to detect faces and keep track of them. It used a method called *depth from focus* which is able to estimate the 3D position of the human using the pan-tilt-zoom of a camera. In 2004, another work [22] also used a stereo pair to obtain images with depth information but improved human segmentation by using a deformable head-shoulder model. In the second half of the twenties, laser scanners started to become an affordable sensor for a service robot and many researchers started to use them. The most common use is to detect legs in order to match these results with the ones from face detection performed in RGB cameras [23]. In the same line, the combination of those results centred the effort of some researchers [24, 25], which produced even more robust solutions. Moreover, there were many researchers which introduced advanced prediction based techniques (Kalman filters, multiple hypothesis trackers,...) in order to detect and track humans using only the laser scanner [26, 27]. Other sensors that can be used for human detection and tracking, include thermal cameras, which are also a popular solution [28, 29] for human detection in certain scenarios.

Many of these human trackers, will probably miss the person being followed if they do not see this person for a short period of time. This happens because they do not maintain a model of the person robust to changes in the environment, such as illumination or pose variations. This represents an important handicap for our tour-guide robot that learns routes from an instructor, because the robot might start following the wrong instructor. In this regard, amongst those works which can actually recognise a human after loosing sigh of it, there are those which carry out face recognition [30, 31], or speaker direction detection [32] to identify the target. Nevertheless those solutions require the target to stand close to the robot, facing it. However, other works [33, 25] avoid these restrictions by recognising humans using the colour of their torso clothes.

2.3 The person following behaviour

The person following behaviour of our tour-guide robot is aimed at reducing the drawbacks that are present in most person-following approaches. For this reason, we have focused our

attention in achieving robustness and adaptability to real world conditions like those of challenging illumination and crowded environments.

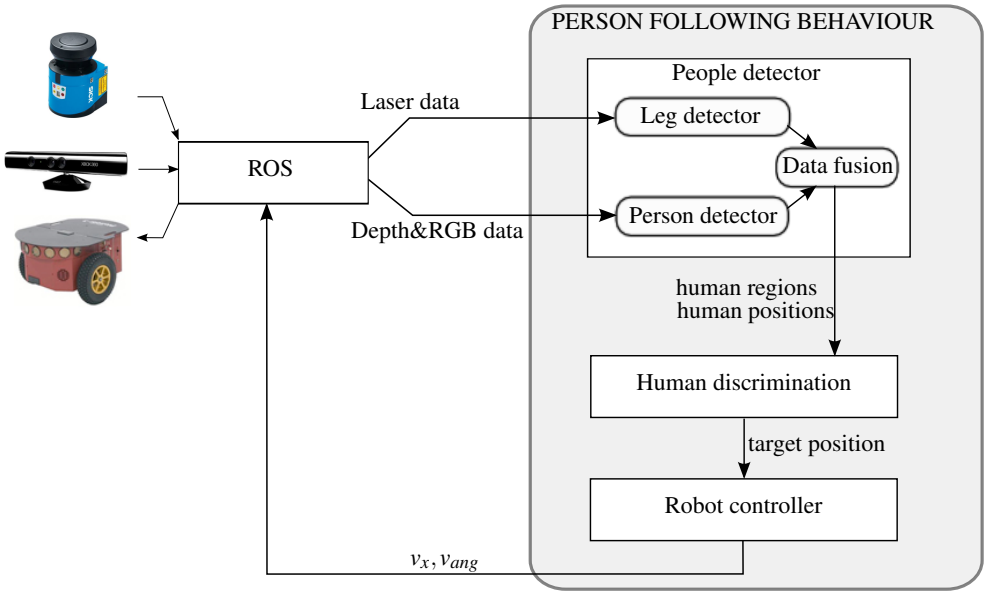


Figure 2.1: Schematic representation of the person following behaviour (shaded in gray) and its interaction with ROS and the robot sensors. The system uses information from two sensors: a RGB-D camera and a laser scanner. Then, it processes this information to obtain the position of the target, and send to the robot the desired angular and linear velocities.

An overview of the system that we have developed to solve the person following task can be seen in Fig. 2.1. In that figure we can see that the person following behaviour (shaded in gray) consists of three main processes:

1. People detector. Contrary to the strategy followed in most related works, we do not look for faces to detect people and extract the corresponding features, but instead we try to detect humans using their body [18, 34]. This fact removes a clear limitation from previous works, the humans who are being followed do not have to face the robot anymore. As we can see in Fig. 2.1, this process contains a laser leg detector and a person detector from camera frames, which are able to work together to provide robust human detections. A description of the different human detectors that we have used in our tour-guide robot is detailed in section 2.4.

2. Human discrimination. It can also be referred as human tracking, recognition or identification, and it refers to the problem of keeping track of a human over time. Most works which address this problem do not consider the possibility of enhancing the discrimination between the person who has more interest for the robot (its target) and the rest (the distractors). We will introduce a novel adaptive strategy that deals with this problem, improving the robustness of the human discrimination task [19]. It consists of dynamically weighting a set of selected visual features according to their discrimination ability in each instant. In this way, it is able to adapt to different environments and to changing illumination conditions. Our solution is similar to those developed by Kuo et al. [35] and Collins et al. [36] since we also deal with the problem of enhancing the differences between two objects. The first one proposes the recognition of different models using AdaBoost, and the second aprroach selects the most discriminative visual features between an object and the background near it. In both cases the goal is to distinguish an object from the background. In our case, we propose a novel approach that is able to select the most discriminative visual features between a set of humans. This process is a core element of our person-following behaviour, and thus, we extensively describe it in section 2.5. There, we can see an study on the set of features that should be used in our robot, as well as a study on how these features can adapt their behaviour to discriminate the robot's target from its distractors.

3. The robot controller. The goal of this controller is to avoid obstacles and to maintain the target at a safe distance and centred in the robot's field of view. In our robot the safe distance is established at around 1 meter, given its maximum speed. This controller is very simple, it is based on the potentials field technique [37], in which the target position exerts an attractive force, while the obstacles around the robot determine the repulsive forces. In the end, it is able to determine the precise linear and angular velocities that the robot must carry out, so that it safely follows the target.

As we have already mentioned, in the following two sections, we describe two studies that have addressed the inherent challenges of the first two processes.

2.4 People detector

The human body is one of the hardest things to characterise because it can vary significantly in pose, size, appearance and its non rigid shape. Despite of that, there are some domains

where highly robust solutions have been proposed, for example, static cameras can rely on background subtraction which makes the recognition process easier. Nevertheless, service robots need to move around complex environments where those solutions are not valid.

In order to provide the tour-guide robot with human detection capabilities, two methods have been analysed [18, 34]. It is important to remark that both methods allow the human to walk freely. In this regard methods based on face detection were discarded because the humans were constrained to walk facing the robot. On the one hand, the first method analysed was a well known general purpose human detector [38] which uses the image gradients to classify regions as human or non-human. On the other hand, the second method was the creation of an own and simple human detector which uses depth images from a RGB-D camera [34]. Nevertheless, as it has already been introduced in the previous section, in order to improve the robustness of either methods, their results are combined with those from a laser leg detector.

2.4.1 Visual human detector

This approach was originally proposed by Dalal & Triggs [38] back in 2005, as the most accurate solution for human detection in grayscale images. Nonetheless, it has received several performance and accuracy improvements over the last years, which have maintained its top position as one of the most versatile and accurate solutions for human detection in grayscale images.

Essentially, the algorithm conceives the human detection problem as an image classification problem, where a support vector machine (SVM [39]) classifies regions of an input image into two classes: human and non-human. In Fig. 2.2, we can see a representation of the two stages that compose Dalal's solution: a) a training stage, in which image features are extracted from both positive and negative samples of humans, and then used to train the SVM classifier, and b) a classification stage, in which the same features are extracted from multiple regions of an input image, and then handed out to the classifier to identify those which contain humans.

Precisely, one of the most important parts of this algorithm are the features used in the process, which were presented in the same work [38]. These features are called histograms of oriented gradients (HOG), and comprise different processing steps over the input image. First, we need to compute the image gradients to obtain both intensity and orientation of those gradients in each pixel in the input image. Then, we need to divide the input image into cells (usually composed of 8x8 pixels), in which we will build an individual orientation histogram

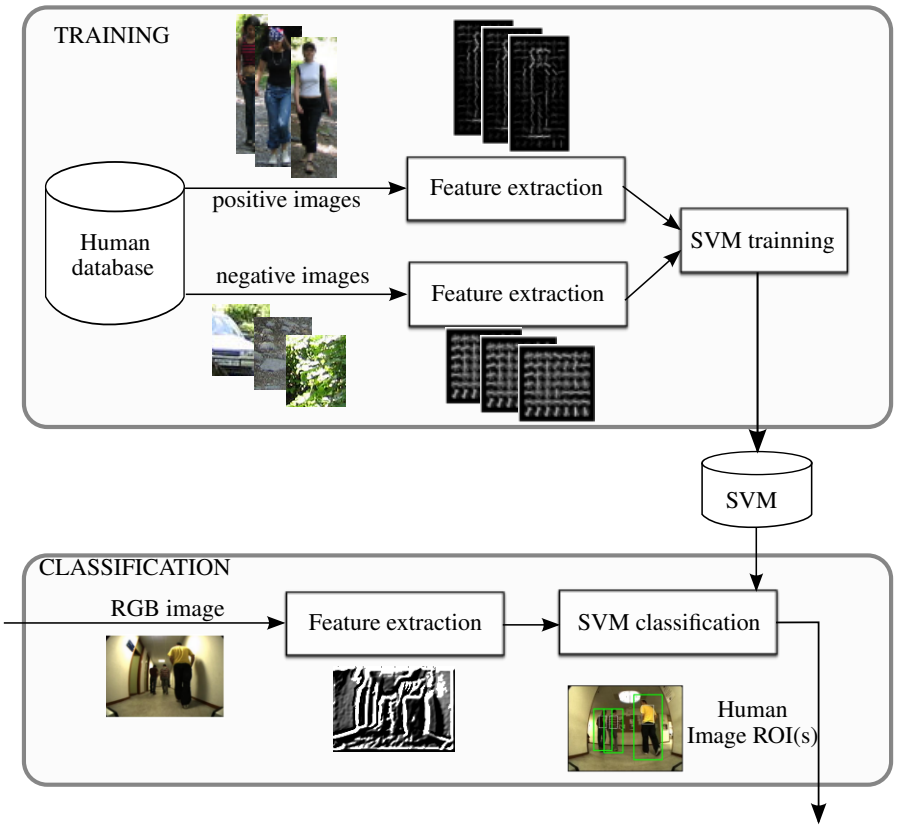
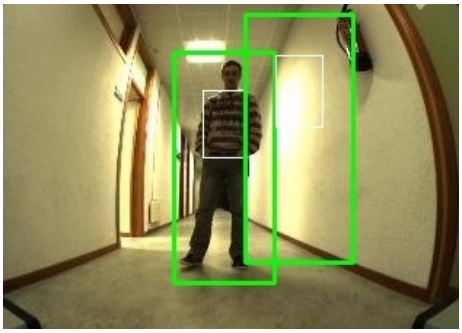


Figure 2.2: Schematic representation of the Dalal & Triggs human detector for RGB cameras. The gray rectangle in the top corresponds to the offline training stage, while the bottom one represents the online classification of image regions into human/non-human.

weighted by the gradient intensities. Typically, the resulting histogram (or HOG) has 9 bins of orientation.

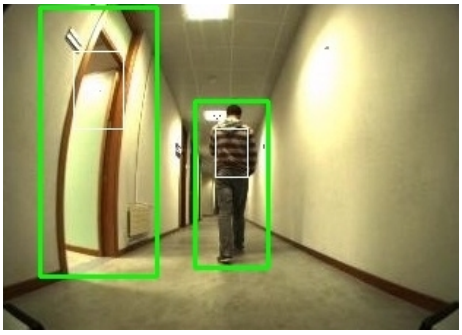
In order to localise the regions of interest (those which contain humans), the algorithm uses the sliding window technique. This technique is widely used in computer vision and consists on the following idea: first, we establish a minimum and maximum scanning region size (i.e. smallest and biggest possible human sizes in the images), then, starting from the top-left image corner, until we reach the bottom-right one, we move the scanning region from left to right and top to bottom. Each time that we move the scanning region, we need to extract the features from that region and ask the classifier if there is a human. In case that we find a



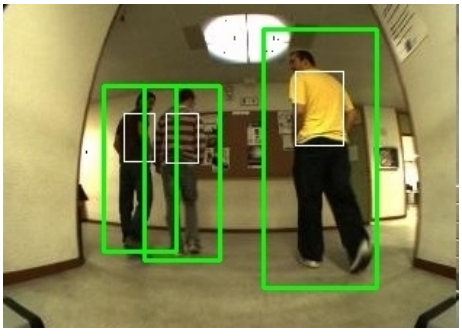
(a) Challenging illumination

(b) Pose variations

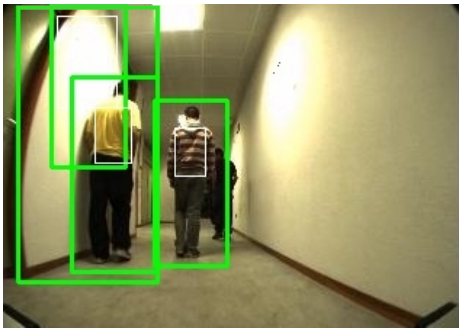
(c) Background interference



(d) Human-like objects



(e) People being correctly detected



(f) True positive and false positive detections

Figure 2.3: Samples of the OpenCV implementation of the HOG detector performing on several images taken from a sequence recorded with the tour-guide robot. The green rectangle represents the region of the image where the classifier has found a human-like object. The inner white one represents the region where the human's torso is expected to be.

positive result, we add this region to a pool. In the end, this pool is processed to find the most probable regions which contain humans, and thus the result of the algorithm are rectangular regions of the image that likely contain humans. In this regard, Fig. 2.3 shows five captures of a sequence taken from a test with our tour-guide robot. More specifically, in Fig. 2.3(a) the lights produce a brightness with human shape in the walls which makes the human detector output a false positive (a green rectangular region over the wall). Fig. 2.3(b) contains two humans that are not detected due to their pose and the illumination conditions. Fig. 2.3(c) shows misplaced regions due to the pose of the humans and the lights in the background. Fig. 2.3(d) shows an example of a door which looks like a human for the human detector. Nevertheless, the detector works well in many situations such as those shown in Fig. 2.3(e).

Finally, we have to remind that we are detecting regions of the image candidates to be humans, and to use those regions to recognise them (Fig. 2.1). When using this algorithm, the torso region is easily segmented by simply down-scaling the original human bounding boxes into smaller ones, as it is illustrated in Fig. 2.3 with the green rectangles being down-scaled to the inner white ones. Moreover, we have decided to use the human's torsos because it is one of the most discriminative parts of a human, and it is visible regardless of the walking direction. This same strategy has been followed by others [33, 25] who also realised that discriminating humans using face recognition from a mobile robot was not a feasible option due to the inherent pose requirements.

Although, one of the major drawbacks of this algorithm is that the original proposal is computationally expensive (processing a 640x480 image can take up to a second), and accordingly not suitable for the real time requirements of a service robot. Nevertheless, the OpenCV [40] project released an implementation of the original algorithm with several speed up improvements (i.e. it included a cascade classifier). In the tests performed in the tour-guide robot a laptop with an 2011 Intel®Core i7 could scan about 15 images per second, which made this implementation suitable for usage in our tour-guide robot.

As a summary, on the one hand, the advantages of using this method are: a) it is a state of the art algorithm in human recognition b) it requires a conventional camera that is available in most service robots and c) there are real-time implementations available. On the other hand, the disadvantages are: a) it does not detect humans in some orientations/poses, b) the segmentation of the torso might not be accurate enough, introducing noise in the characterisation of the humans and c) it requires the humans to be fully visible in the camera images: bad performance with occlusions.

2.4.2 Human detector for depth images

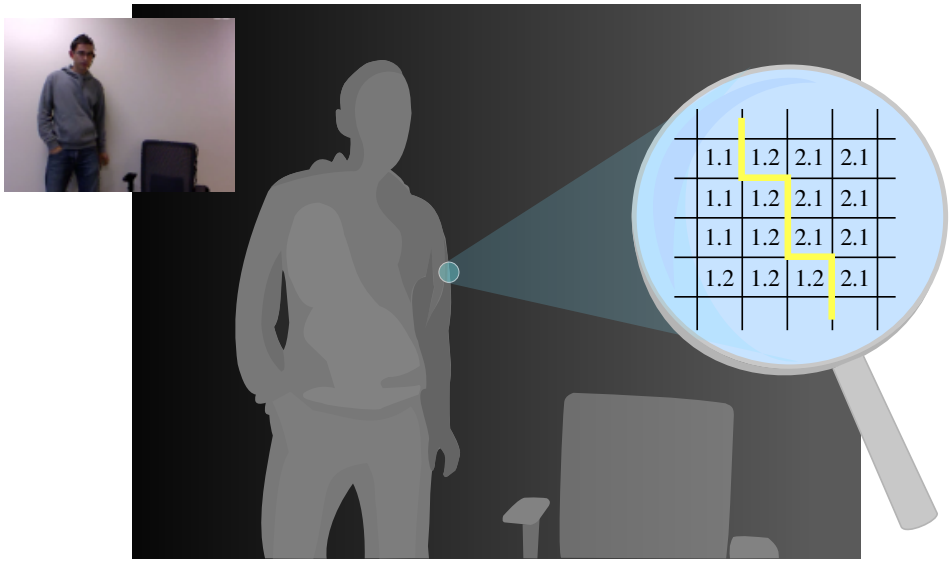
The Microsoft Kinect, a consumer electronics device first intended for a gaming console, revolutionised the robotics community by offering a RGB-D camera for about 10 times less the cost of previous ones. Essentially, this device consists of two components, a RGB camera aligned with a depth sensor from PrimeSense[1]. The depth sensor implements a technology known as structured light, which is able to compute the distance from the camera to the real world in each pixel of an infrared image. In order to create this image, a projector illuminates the scene with a special infrared pattern, and measures the distortion of this pattern when projected on objetcs' surfaces. Depth can eb computed from the deformation data. Therefore, the Kinect provides our tour-guide robot with a colour image (RGB camera) aligned with a second depth image (PrimeSense Sensor), or in other words: an affordable RGB-D sensor.

We have to remind that the regions of the human body are the inputs of the next stage of our person-following behaviour (Fig. 2.1). So, our hypothesis is that a human detector which uses depth information will allow the precise segmentation of the human body from the background, and thus a better characterisation of the human will take place in the next stage of our person-following behaviour. In this regard, we have designed a simple, and effective human detector algorithm for depth images [34].

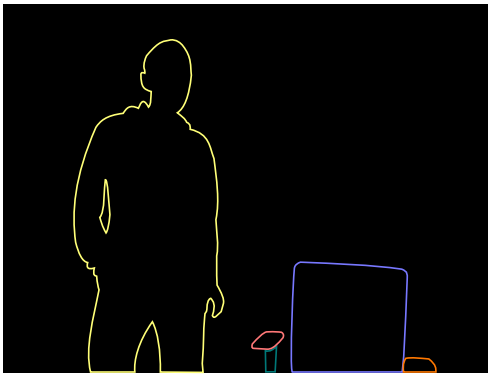
The main goal of our human detection algorithm is to detect stand-up humans from a mobile platform, regardless of the orientation. For this reason we can not use of background subtraction or common motion detection techniques. In the same way, strategies that look for certain parts of the human body also had to be discarded: a face detector or a head-shoulders detector would not work when those parts are occluded due to the orientation of the human. Instead, our human detector algorithm relies exclusively on depth images, and the concept of *depth differences*. That is, we assume that two humans are not touching each other or the environment (walls, doors), and thus we proceed as follows:

1. Pre-processing the depth image. Before applying the next steps of our algorithm we need to remove the floor and low precision data from the depth image. The floor is erased by computing the height of each pixel in the image, and removing those which are under a threshold (usually 10cm from the ground). The height is computed by transforming the depth image into a point cloud and subtracting the height at which the sensor has been put in our robot. Regarding low precision data, the Kinect has an

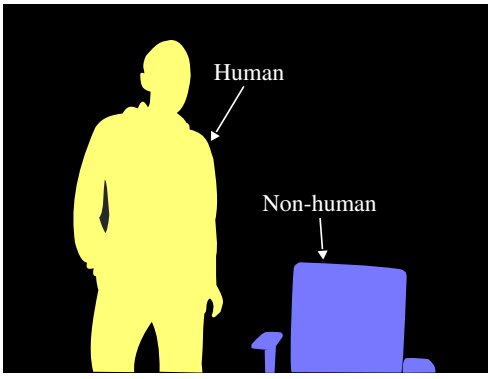
[1] http://www.primesense.com/solutions/3d-sensor/



(a) Scanning a depth image for significant differents in depth data



(b) Image segmented in multiple objects



(c) Merging of close objects and classification

Figure 2.4: Different stages in our human detector algorithm for depth images. The algorithm starts searching for big depth different amongst adjacent pixels (a). Then, it connects those regions defining the contours of multiple objects (b). Finally, the algorithm merges the contours which might be part of another to create big objects (c), and selects only those which meet the properties of a human (size, shape, etc).

average error of about 1.5cm in the range 0.5-3.5m but this error increases exponentially from 3.5m to 7m to reach values of over 0.1m. Therefore, we also remove data from the depth image which is above 4 meters.

2. Depth difference segmentation. In Fig. 2.4(a) we can see an illustration of a depth image and its corresponding RGB image. The depth image is coloured using a gray scale, where black represents a far distance and white a close one. For each pixel of the depth image we compare its value with the values of its adjacent pixels at its right, top-right, bottom-right and bottom. If in any of those comparisons we find that there is a difference in depth higher than a threshold, we mark that pixel as being part of the contour of an object. We have illustrated this step with the magnifying glass of Fig. 2.4(a). The result of this process are the contours of multiple objects in the image as we illustrate in Fig. 2.4(b).

3. Merging of broken objects. There are some objects that due to its pose or orientation might be segmented in several smaller objects as a result of the process described in the previous step. An example of this segmentation can be seen in Fig. 2.4(b) with the chair, where the armrests where identified as individual objects. In this regard, we compute the positions of every small object identified in the previous step, and if they are close enough (we use a threshold of 0.3m) to a big object we join them together. Fig. 2.4(c) illustrates this process with an example, where the different parts of the chair were put together.

4. Classification of objects. In this final stage, we compute several properties of each object such as height, average width and area. Using these properties we classify the objects into human or non-human. In the example of Fig. 2.4(c), we filter out the chair for being too wide for such as short height.

It is obvious that an object with a volume and shape of a human can be misclassified, fortunately, we have found very few objects in real world experiments with these properties, and even when one is misclassified it did not represent a real problem for the tracking of the actual target of our robot. Our human recognition approach will detect that it is a distractor and not the robot's target. An example of an object which might be misclassified is a clothes hanger.

This algorithm represented many improvements over the visual human detector that we were using, but also brought us some drawbacks that should be solved with future improve-

(a) Two humans in different poses

(b) A poster correctly filtered as non-human

(c) Three humans with partial occlusions

(d) Three humans in different poses

(e) People is detected up to a level of occlusion

(f) Two humans are detected as one (green)

Figure 2.5: Different examples of our human detector for depth images. The samples are taken from different scenes recorded in several demonstrations where we have showcased our tour-guide robot. A human is detected only if his body is shaded with some colour, and his position drawn in front.

ments on the algorithm. On the one hand, the biggest advantages of using this method are: a) it requires small computational power to process the depth data provided by a low cost depth sensor and thus it can be used in low cost robots, b) to analyse a depth image, the algorithm uses less than 10ms on a 2012 Intel®Core i7 laptop, c) it is able to detect a person regardless of their orientation with the camera (Fig. 2.5), d) it is able to detect humans partially occluded by other people (Fig. figs. 2.5(c) to 2.5(e)), or when they are close to the robot and only part of their body is visible, and e) it is able to reach pixel level segmentation of the human body, leaving most of the background outside the detected region. On the other hand, the major drawback of this algorithm relies on the fact that if two persons are touching each other it will not detect them as a human or in some cases both persons are detected as if they were only one as illustrated in Fig. 2.5(f).

After having experimentally tested both methods, we selected the second method for our tour-guide robot because of the aforementioned advantages, being the precise segmentation of the human body one of the most relevant. This will let us improve the accuracy of the next stage of our person-following behaviour, human identification and tracking (section 2.5).

2.4.3 The laser leg detector

As we have shown in Fig. 2.1, the human detector in our person following behaviour contains a camera detector, which we have just described but also a laser leg detector algorithm. The role of this second one is to improve the robustness of the people detection by combining the results from both algorithms [18, 19, 34].

The laser leg detection algorithm that we have implemented in our tour-guide robot is based on the a work by Gockley et al. [41]. Essentially, it works in a similar way than our human detector from depth images:

1. First, we need to pre-process the laser data to remove noise. It is quite frequent that a part of laser beam shoots the edge of an object, while the other part shoots the background. When this happens, the computed value is the average of both distances, thus obtaining noisy data.

2. Then, we break up the laser data into segments. The start and end points of a segment are defined by the points where there is a depth difference higher than a threshold (we set it to 0.1m).

3. Next, segments are classified, based on their size, into legs or non-legs. According to laser resolution, we set a minimum width for a leg at 0.05meters and the maximum at 0.45m, which corresponds to the case where there is no separation between the legs of the human.

4. Finally, we group legs into pairs of legs in case there are two thin legs close to each other.

The results of this algorithm are the positions of the detected pairs of legs.

2.4.4 Increasing the field of view of the camera

The field of view of the camera is much more narrow than that of the laser. This might be a problem when the robot is following a human who moves a bit faster than the robot's chances of correcting its position. For this reason we have decided to include an small actuator to horizontally rotate the robot's camera towards the position of the human that the robot is following.

Figure 2.6: The Dynamixel AX-12A installed on top of the mast that holds the camera. This configuration allow us to pan the robot's camera.

Fig. 2.6 shows a picture of the installation that we have made to attach a Dynamixel AX-12A on top of the mast that holds the camera in our robot. The goal of the module that

controls this actuator is to horizontally rotate the camera towards the angle where the target was previously detected. It will continuously try to keep the human in the centre of the image to reduce the possibilities of loosing sight of him. This configuration allows the robot to rotate the Kinect from about -70 to 70 degrees, where 0 is the heading of the robot. In practice, with this solution we obtain an increased field of view for the camera close to that of the laser scanner.

2.4.5 Sensor fusion in the human detector

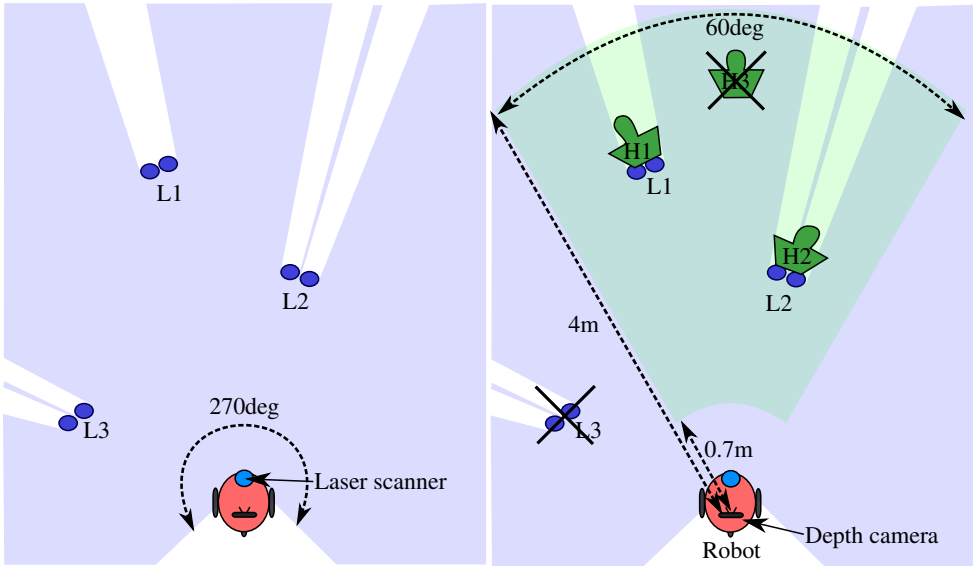


Figure 2.7: Illustration of the fusion of the results from the laser leg detection algorithm and the depth human detector in our tour-guide robot.

The human detector from depth images and the laser leg detection algorithm contain classifiers which have been designed to get a high percentage of detections in spite of introducing some false positives. In this way, we make sure of obtaining most of the actual humans, relying on the ability to remove the false positives by combining the results from the laser leg detector with those from the human detector in depth images. Fig. 2.7 illustrates the sensor fusion process with one example. In the left side of the image, we can see how the laser leg detector has found three pairs of legs located at the positions L_1, L_2 and L_3. In the right side of

the image, our human detector for depth images detects three humans located at the positions H_1, H_2 and H_3. Therefore, a simple matching of each L_n with each H_m position is able to discard those humans without correspondence in one of the detectors. In this matching we use a threshold of 0.5 meters. In the end, this process leaves the tour-guide robot with two robust detections H_1 and H_2.

Note that, since the camera might be rotated, we need to previously rotate the (x,y) coordinates of the H positions, to obtain the new (x',y'):

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned} \tag{2.1}$$

where θ is the camera angle as provided by the Dynamixel Ax-12A actuator.

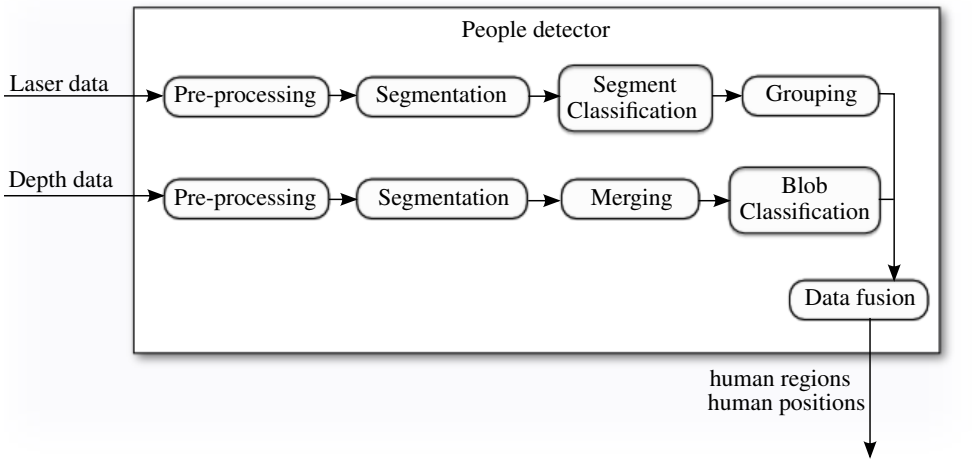


Figure 2.8: Flowchart of the full people detection process that takes place in our tour-guide robot.

To summarise, the people detection process is illustrated in the flowchart of Fig. 2.8. It has proved to be very effective in the tests that we have carried out with our tour-guide robot, as we have shown with the samples of Fig. 2.5. Moreover, the whole human detection task takes no more than 15ms in a 2012 Intel®Core i7 laptop, which leaves enough processing power for other tasks in our robot. This point is very important in any service robot, since they usually have to deal with several problems at the same time. Precisely, in the next section we describe how our robot identifies the humans our robot detects.

2.5 Human discrimination

Human discrimination in service robots is also termed as human identification, human recognition or human tracking. These designations usually refer to the problem of identifying the humans in the robot's surroundings, i.e. which one should a robot follow, avoiding mistaking him with a distractor[2].

The discrimination between the robot's target and its distractors is a critical task in our robot (as we have already introduced in Fig. 2.1), but also for any other robot that has to follow humans or to simply deal with them. For this reason, in this thesis we have made a considerable effort to obtain a robust human discrimination solution, that is able to cope with changes in illumination and temporary occlusions of the target.

Most person tracking methods are based on position predictors, which prevents them to properly handle general types of occlusions and movements. Although we are not the first to deal with these problems (in section 2.2 we refer some recent works that have explored the possibilities of using colour features to overcome occlusions), we believe that using a descriptor that takes into account their previous position, as well as, an appropriate set of visual features will allow us to keep track of a human in a wider range of situations than previous works.

We understand the discrimination of the target from the distractors as a matching problem, where $R = \{r_1, r_2, ..., r_n\}$ is the set of detected human regions (as given by the human detector), and $H = \{h_1^t, h_2, ..., h_n\}$ is the set of unique humans that our robot has recently seen, where h_1^t is the robot's target. In this regard, our task is to correctly match each of the human regions in R into its corresponding human in H. Moreover, in the case of our tour-guide robot, we should be specially careful when dealing with h_1^t, avoiding the matching unless the probability is high.

Therefore, given that in our human discrimination algorithm we use a descriptor containing two different kinds of features (position and appearance), the probability of a human region r of matching the instance of a human h is defined by Eq. 2.2:

$$P(h|r) = \frac{P_p + P_a}{2} \tag{2.2}$$

[2]The distractors are people present in the environment where the robot operates, distinct from the robot's target

where P_p is the probability according to the position, as defined by Eq. 2.3):

$$P_p = \begin{cases} 1 & \text{if } d_p \leq 0.3 \\ \frac{-d_p}{0.7} + \frac{1}{0.7} & \text{if } d_p \geq 0.3 \ \& \ d_p \leq 1 \end{cases} \tag{2.3}$$

where d_p is the distance in meters between r and h.

P_a is the probability according to the appearance, or similarity between r and h, which is defined by Eq. 2.4:

$$P_a = similarity = \frac{1}{n} \sum_{f=1}^{n} d_f \quad \in [0,1] \tag{2.4}$$

where n is the number of visual features that are compared, and d_f is the Bhattacharyya distance between the histograms of that feature f in the human region, $hist_f^r$, and the histograms of that same feature in the visual descriptor $hist_f^h$, as defined by Eq. 2.5:

$$d_f = \sqrt{\sum_{i=0}^{b-1} \sqrt{hist_f^r(i) hist_f^h(i)}} \tag{2.5}$$

where b is the number of bins in those histograms. The comparison of visual features using histograms is common in similar works [33, 25], since it brings many advantages: it is invariant to the size of the human region, and to the pose of the human. That is, it will not reduce its similarity measure when the human moves towards or away from the robot, and it will not reduce its similarity when the human rotates or changes its pose.

Finally, a region $r \in R$ matches the human $h*$ when it maximises $P(h|r)$ and is above a threshold, that is:

$$h^* = \arg\max_h P(h|r) \tag{2.6}$$

We establish the threshold $P(h|r) \geq 0.5$ when it comes to match any human distinct from the target, and the threshold $P(h^t|r) \geq 0.8$ when the matched human is the target. Finally, once we have classified the human regions we will update every h with the properties of its matched r, assigning the new position and appearance to that human.

In this point, it is obvious that the visual features play an important role on a correct discrimination between the robot's target and the distractors. For this reason, we have carried out an in-depth search on which ones would be the most suitable set of visual features for our visual descriptor [19]. In section 2.5.1 we will describe this process, which will result on eight visual features that provide robust and non-redundant information at low computational

cost. In addition, in section 2.5.2 we propose a novel algorithm that is able to dynamically adapt this set of visual features to further improve the discrimination between the target and the distractors in challenging situations [19].

2.5.1 Determining the visual features for human discrimination

The visual appearance of an object can be described in terms of features such as color, texture, edge density, gradient magnitude and so on. However, many of these features are redundant (i.e. colour spaces), and computing all of them is not a feasible option for our tour-guide robot. Therefore, the first problem that we have to tackle is to determine which features should be included in our visual representation of the human (the descriptor).

In order to do so, we have proceed in two parts. First, in part A, we have selected a initial *pool of features*, and then, in part B, we have gone through a selection process of the most appropriate set of features for our robot.

Part A: Initial pool of features

The pool of features that we have evaluated is composed of features that have shown good performance in object recognition or pattern classification problems.

There are some popular features that we had to discard due to the constraints inherent to our problem. The most important is related to the real-time computational requirements of any robot. Our goal is to be able to extract the features of our human model in less than 100ms for all the humans in the robot's field of view. In this regard, some works use features that derive from Fourier, Gabor or Radon [42] transforms, these approaches require the computation of statistical properties from many filtering processes of camera frames, which result on high computational requirements.

Other popular features that we have discarded are SIFT and SURF features, because their computational cost [43, 44] is far from our requirements. In general it seems to be reasonable to start trying with a set of simple (from the computational point of view) features, and analyse whether the performance achieved with them is good enough. In addition, another important aspect that must be considered is the fact that we will establish a comparison amongst the features of our initial pool (part B: the selection process), in order to select those that provide the biggest quantity of novel information about a human. As we will explain, this can easily be done with mutual information when the features are *single valued*. Taking into account

these two restrictions, the following is a description of the 27 features in the initial pool of features:

– *$H_1L_1S_1$, L_2AB, YCbCr, H_2S_2V, and Grey*. These are all common representations of colour. Since hue, lightness and saturation are features that appear in several of the colour spaces, we have added subscripts to differentiate them in our analysis.

– *Local Binary Patterns (LBP)*. The classic LBP [45] describes the local shapes contained on a gray-scale image. For every pixel of a gray-level image, the value of the LBP is defined by two parameters: P, the number of neighbouring pixels that are taken into account to compute the LBP (usually 8, 16 or 32), and R, the radius of the circle that contains those P pixels (usually a 1,2 or 3). For our study, we have selected the $LBP_{P=8,R=1}$ because it is the most popular, and it is also the LBP with the lowest computational cost (it only considers the 8-neighbours of each pixel).

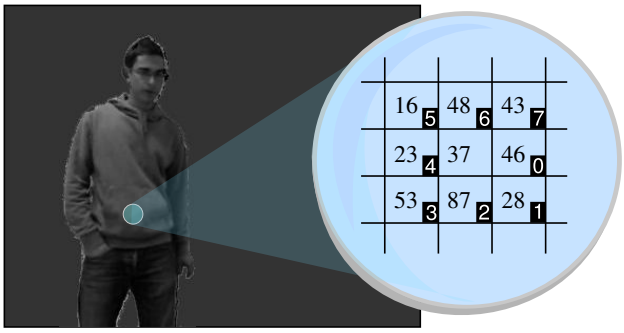
  The process of computing the $LBP_{8,1}$ in a pixel g_c is illustrated in the example of Fig. 2.9. First, we have to select the eight neighbours of g_c, assigning them a p index as illustrated in Fig. 2.9(a), where the number in the black box indicates this index. Then, we compute the gray-level differences between g_c and every g_p (Fig. 2.9(b)). Next, we perform a thresholding (Eq. 2.8) to obtain a binary code in the order indicated by the p indexes (Fig. 2.9(c) and find its decimal value: 179, which is our value of the $LBP_{P=8,R=1}$ at g_c. This process can also be described with Eq. 2.7:

$$LBP_{8,1} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{2.7}$$

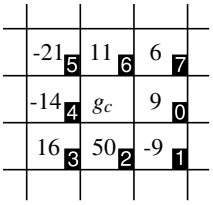
  where $s(x)$ is the threshold function:

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.8}$$

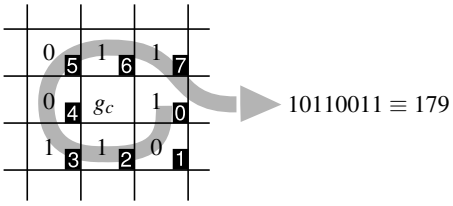
– *Census-LBP (Census)*. This feature is based on the census transform [46]. Similarly to LBP, this feature also computes a binary code following the procedure shown in Fig. 2.9. However, the order of bits is different: the binary code is built from left to right starting in the top row and finishing with the bottom row, as illustrated in Fig. 2.10. Thus, if we use the same example as in Fig. 2.9, the binary code would be 01101110 and 110 would be the corresponding decimal representation assigned to the central pixel (g_c).



(a) Scanning a human region extracting 3x3 pixel regions



(b) Difference operator



(c) Thresholding, code extraction and final value

Figure 2.9: Process to extract the $LBP_{8,1}$ features from the gray scale images of the human regions. This process has to be repeated for every pixel g_c of the human region.

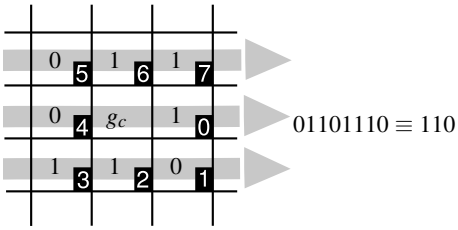


Figure 2.10: Binary code extraction and final value of the censusLBP. The previous stages do not differ from those in the classic LBP.

– *Centre-Symmetric-LBP (csLBP)*. Heikkila et al.[47] presented this feature as a combination of the strengths of the well-known SIFT descriptor with those of the LBP features. It is a robust and computationally efficient feature with tolerance to illumination changes.

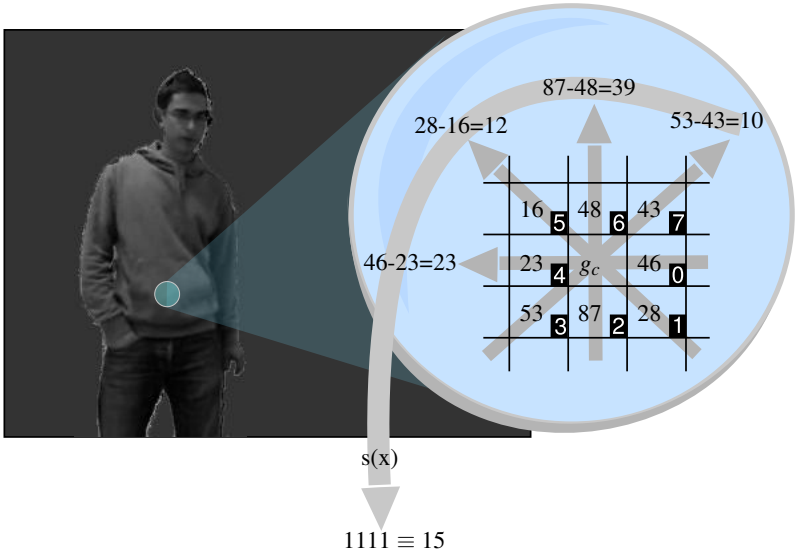
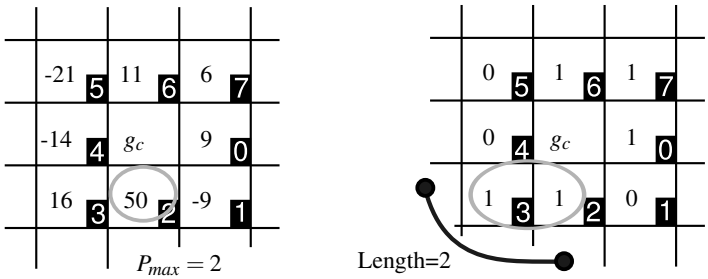


Figure 2.11: Process to extract the csLBP features from the gray scale images of the human regions. This process has to be repeated for every pixel g_c of the human region.

Fig. 2.11 summarises the procedure that we have followed to compute this feature. This feature is based on the gradient of the gray level along four directions (the arrows). Therefore, we compute gray-level differences along these four directions, and threshold the results using Eq. 2.8. As a result of this process we obtain a binary code of four digits. The decimal representation of the binary code is the value of the CSLBP assigned to the central pixel. In the example of Fig. 2.11 this value is 15.

– *Semantic-Local Binary Patterns (SLBP)*. The S-LBP were proposed by Mu et al.[48] to detect humans in images. This variant of the classic LBP improves its semantic consistency. In the proposal developed by Mu et al. each pixel of the original image can have more than one value associated, to avoid this we slightly modified the original

proposal. In this regard, the following are the steps that we carry out to compute this feature:



(a) Difference operator and p_{max} (b) Counting the arc length

Figure 2.12: Extracting the value of p_{max} and the length of its arc, which are needed to compute the value for the SLBP feature.

Since this feature is similar to $LBP_{8,1}$, we will use the same example and naming that in Fig. 2.9(a). In the first step we also compute the differences between g_c and each g_p. However, the next steps are different. Now, we have to search for the p index of the maximum absolute difference (which we call p_{max}), which in the case of our example is located at $p = 2$ (Fig. 2.12(a)). Then, we threshold (Fig. 2.12(b)) the differences in the same way as we did to calculate the LBP binary code (Eq. 2.8). Next, using this binary code, we need to measure the $length$ of the arch which contains p_{max}. An arch in the binary code corresponds two a sequence of ones. In the example of Fig. 2.12(b), the pixel with p-index 2 belongs to an arch with also length two. Notice that the length of this arc might be zero, if the gray level difference in p_{max} is negative (the pixel might not belong to any arch). After obtaining these values we combine them to obtain the value of our S-LBP according to:

$$SLBP_{8,1} = 8p_{max} + length - 1 \tag{2.9}$$

In the example illustrated in Fig. 2.12, the value of the S-LBP for the pixel at the centre is 17. The values for this feature range from 0 to 63.

– *Edge density (ED)*. This is a texture descriptor which measures the density of the edges in an image. In order to compute it, the first step is to apply any edge detector, for this

purpose, we use the Canny edge detector [49], although any other could be used. The resulting image will contain ones in those pixels where an edge has been found and zeros otherwise. Once this has been done, the feature value (edge density) for a given pixel x, is computed as the sum of the ones included in a 9×9 window centred on the pixel x. In the case that the sum of ones in the aforementioned window is higher than 31, the transformed value of the pixel will still be 31. This limit was set heuristically after checking that it was quite improbable to find a region where more than the 38% of the pixels belonged to edges, and thus reduce the sparseness of the feature histogram. To sum up, we believe that this feature is a good descriptor of the amount of texture in the neighbourhood of a pixel, or in other words, a descriptor of how the texture is distributed in the image.

– *Gradient orientation (GO)*. More commonly known as Histogram of Oriented Gradients (HOG) (Dalal et al. [38]), this is one of the most popular features in the last years due to its high performance in object recognition. We have used this features to detect humans (section 2.4.1).

  The original feature consisted in the computation of a orientation histogram inside image cells, this resulted on a multi-valued feature. In order to fit this study, we have removed the divisions of the image (for more details refer to the original implementation from Dalal et al. [38]), but we have kept the original idea of combining orientation and gradient magnitude in a single histogram. More specifically, we classify the magnitude and orientation values into a eight-bin magnitude histogram and a eight-bin orientation histogram, respectively. Then, we compute the orientation bin (b_o) and magnitude bin (b_m) of each pixel, and combine them into a unique number for every pixel of the original image (Eq. 2.10):

$$GO = 8b_m + b_o \tag{2.10}$$

  Since the number of bins used to build the histogram is within the interval $[0,7]$, the values that this feature take in every pixel go from 0 to 63.

– *Edge descriptor (EHD)*. This feature is inspired on the MPEG-7 Edge Histogram Descriptor [50]. The EHD represents the distribution of five types of edges in sixteen sub-images of the same size.

In our case, we compute the distribution of those five types of edges in the whole image. For this purpose, in every pixel of the original image, we perform the five filter operations proposed in the original algorithm. In case that the filter response is higher than a threshold, it would mean that we have found one of the edges that the algorithm searches for (horizontal, vertical, 45 degree diagonal, 135 degree diagonal or non-directional). Thus, we assign a value from 1 to 5 to the pixel at the centre, depending on which edge was found. In the event that there is no edge (low response for all the filter operations), the pixel will get a zero value.

– *First and second gray level derivatives ($\partial x, \partial y, \partial^2 x, \partial^2 y$)* along the horizontal and vertical axes. These are also very well-known descriptors used in many computer vision tasks such as in human tracking tasks [35].

This feature pool covers most of the features that have shown good performance in related tasks, and therefore it is a good start point for the next phase: the *selection of the set of visual features*.

Part B: Selection of visual features

Computing all the features that we have just described is not a feasible option for a mobile robot with limited computational capabilities like the one we are developing. Therefore, the second part of our study consists in performing a feature selection over the initial pool of features. Feature selection is the process of selecting the most appropriate subset of features to accomplish a classification task. This has been a recursive problem in many fields, with most works presenting different solutions to specific problems [51].

Since, two of the most popular criterias which are being used to select features are: relevancy to the problem (a feature contain useful information for the task) and redundancy (a feature provides different information those we have already selected), we select our subset of features by combining these two criterias.

More specifically, we a) rank the features according to their redundancy by using a technique adapted from the Minimum Redundancy-Maximum Relevance algorithm (mRMR [52]) and b) select the most appropriate number of features according to the f-score. The Minimum Redundancy-Maximum Relevance criterion uses Mutual Information[53] to measure both redundancy amongst features and relevance to the task. However, we will only measure the

redundancy because in this case there is little or none prior information about the target, and thus we can not estimate the *a priori* relevance to the task with this method.

The process of selecting the set of visual features that will describe the human's appearance in our robot is divided in three stages. First the computation of the mutual information (stage B.1), second the ranking of the features according to the redundancy they share with others (stage B.2), and third the selection of the appropriate number of features to build the descriptor (stage B.3).

Stage B.1: the mutual information

In the first stage, in order to measure the redundancy of two features we will use a technique described in the mRMR algorithm [52], which is based on the mutual information between the features.

The mutual information amongst two features X and Y is a quantity that measures how much can be said about the value of one of them, once the value of the other is known. It is defined as:

$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log(\frac{p(x,y)}{p_1(x)p_2(y)}) \tag{2.11}$$

where $p_1(x)$ is the probability of feature X taking the value x, $p_2(y)$ is the probability of getting the value y in feature Y and $p(x,y)$ is the joint probability of getting $X = x$ and $Y = y$ at the same pixel in the image (for example, the probability of $hog = 36$ when $edge = 2$).

In order to compute these probabilities for all the texture and colour features, we need a large database of images. For this reason, we have used the INRIA Person dataset [38]. This dataset is one of the largest, containing several hundreds of human images wearing random clothes in different environments. However, we believe that any dataset, providing that it is sufficiently large, would be valid to compute the aforementioned probabilities and therefore the mutual information amongst different features. The mutual information between all of the 27 features makes up a big table, therefore, for space reasons the results are divided into three tables:

First, table 2.1 shows the mutual information between the colour features. As it could be expected, features which describe illumination (L_1, L_2 and *Grey*) share a high mutual information. The same results are visible between saturation features, S_1 and S_2, and also between hue features H_1 and H_2, which share certain information with A, B, C_r, C_b. When $X = Y$, that is, when the mutual information is computed amongst samples of the same feature,

	H_1	L_1	S_1	L_2	A	B	Y	Cr	Cb	H_2	S_2	V	Red	Green	Blue	Grey
H_1	**5.52**	0.14	0.31	0.17	1.17	1.25	0.16	1.26	1.25	5.29	0.30	0.14	0.16	0.19	0.16	0.16
L_1	0.14	**5.80**	0.20	3.00	0.13	0.13	3.15	0.10	0.12	0.14	0.34	2.68	1.95	2.55	2.04	2.72
S_1	0.31	0.20	**5.10**	0.18	0.38	0.78	0.18	0.48	0.74	0.31	2.80	0.24	0.35	0.19	0.20	0.16
L_2	0.17	3.00	0.18	**5.87**	0.13	0.14	4.52	0.10	0.14	0.17	0.25	2.90	1.57	3.75	2.20	3.41
A	1.17	0.13	0.38	0.13	**2.95**	0.30	0.12	0.66	0.26	1.17	0.44	0.11	0.14	0.14	0.10	0.13
B	1.25	0.13	0.78	0.14	0.30	**3.67**	0.12	0.45	2.56	1.25	0.73	0.15	0.15	0.09	0.12	0.12
Y	0.16	3.15	0.18	4.52	0.12	0.12	**5.82**	0.10	0.13	0.16	0.27	2.86	1.61	3.38	2.30	3.60
Cr	1.26	0.10	0.48	0.10	0.66	0.45	0.10	**3.19**	0.49	1.26	0.48	0.12	0.11	0.09	0.12	0.10
Cb	1.25	0.12	0.74	0.14	0.26	2.56	0.13	0.49	**3.57**	1.26	0.63	0.16	0.15	0.10	0.12	0.13
H_2	5.29	0.14	0.31	0.17	1.17	1.25	0.16	1.26	1.26	**5.52**	0.30	0.14	0.16	0.18	0.16	0.16
S_2	0.30	0.34	2.80	0.25	0.44	0.73	0.27	0.48	0.63	0.30	**5.41**	0.20	0.51	0.26	0.21	0.26
V	0.14	2.68	0.24	2.90	0.11	0.15	2.86	0.12	0.16	0.14	0.20	**5.84**	2.20	2.77	3.49	2.44
Red	0.16	1.95	0.35	1.57	0.14	0.15	1.61	0.11	0.15	0.16	0.51	2.20	**5.83**	1.58	1.11	1.52
Green	0.19	2.55	0.19	3.75	0.14	0.09	3.38	0.09	0.10	0.18	0.26	2.77	1.58	**5.83**	1.87	2.86
Blue	0.16	2.04	0.20	2.20	0.10	0.12	2.30	0.12	0.12	0.16	0.21	3.49	1.11	1.87	**5.84**	2.09
Grey	0.16	2.72	0.16	3.41	0.13	0.12	3.60	0.10	0.13	0.16	0.26	2.44	1.52	2.86	2.09	**5.81**

Table 2.1: Mutual information between the colour features analysed. Numbers in bold indicate the entropy of the feature.

the result is the entropy of the variable. The entropy is the average unpredictability in a random variable, which is equivalent to its information content. Therefore, a high value of entropy is an indicative that the feature has many different values within the dataset.

	SLBP	LBP	ED	GO	EHD	∂x	∂y	Census	csLBP	$\partial^2 x$	$\partial^2 y$
SLBP	**5.1**	3.46	0.10	0.91	0.44	0.35	0.35	3.03	1.20	0.10	0.10
LBP	3.46	**5.66**	0.08	1.64	1.07	0.52	0.50	5.15	2.05	0.17	0.17
ED	0.10	0.08	**3.04**	0.05	0.02	0.28	0.26	0.07	0.01	0.24	0.22
GO	0.91	1.64	0.05	**5.29**	0.78	0.51	0.59	1.53	1.60	0.13	0.16
EHD	0.44	1.07	0.02	0.78	**2.28**	0.27	0.26	0.96	0.83	0.07	0.06
∂x	0.35	0.52	0.28	0.51	0.27	**3.98**	0.14	0.50	0.31	1.24	0.09
∂y	0.35	0.50	0.26	0.59	0.26	0.14	**4.06**	0.47	0.26	0.09	1.23
Census	3.03	5.15	0.07	1.53	0.96	0.50	0.47	**5.15**	1.90	0.16	0.15
csLBP	1.20	2.05	0.01	1.60	0.83	0.31	0.26	1.90	**3.11**	0.08	0.06
$\partial^2 x$	0.10	0.17	0.24	0.13	0.07	1.24	0.09	0.16	0.08	**2.97**	0.10
$\partial^2 y$	0.10	0.17	0.22	0.16	0.06	0.09	1.23	0.15	0.06	0.10	**3.05**

Table 2.2: Mutual information between the texture features analysed.Numbers in bold indicate the entropy of the feature.

Second, table 2.2 shows the mutual information between the texture features. The features based on local binary patterns such as *SLBP*, *LBP*, *Census* and *CSLBP* share a high mutual information. The same fact can be appreciated between the first and second derivatives in each direction.

	H_1	L_1	S_1	L_2	A	B	Y	Cr	Cb	H_2	S_2	V	Red	Green	Blue	Grey
SLBP	0.03	0.07	0.02	0.07	0.02	0.02	0.07	0.01	0.03	0.03	0.02	0.06	0.05	0.07	0.06	0.07
LBP	0.03	0.05	0.01	0.05	0.02	0.02	0.05	0.01	0.02	0.03	0.02	0.05	0.04	0.05	0.05	0.06
ED	0.02	0.02	0.00	0.02	0.02	0.02	0.02	0.00	0.02	0.02	0.00	0.02	0.01	0.02	0.02	0.03
GO	0.01	0.03	0.01	0.03	0.01	0.01	0.03	0.00	0.01	0.01	0.01	0.02	0.02	0.03	0.02	0.03
EHD	0.02	0.05	0.02	0.04	0.01	0.01	0.05	0.01	0.01	0.02	0.02	0.04	0.04	0.04	0.04	0.05
∂x	0.03	0.04	0.02	0.04	0.02	0.03	0.04	0.01	0.03	0.03	0.02	0.03	0.03	0.04	0.03	0.05
∂y	0.03	0.04	0.01	0.05	0.02	0.02	0.05	0.00	0.03	0.03	0.01	0.04	0.03	0.05	0.04	0.06
Census	0.02	0.05	0.01	0.05	0.02	0.02	0.05	0.01	0.02	0.02	0.01	0.04	0.04	0.05	0.04	0.05
csLBP	0.00	0.02	0.00	0.02	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.01	0.01	0.02	0.01	0.02
$\partial^2 x$	0.02	0.03	0.01	0.02	0.02	0.02	0.03	0.00	0.02	0.02	0.01	0.02	0.02	0.03	0.02	0.03
$\partial^2 y$	0.02	0.03	0.01	0.04	0.02	0.01	0.04	0.00	0.01	0.02	0.00	0.03	0.02	0.04	0.03	0.04

Table 2.3: Mutual information between the texture and the colour features analysed.

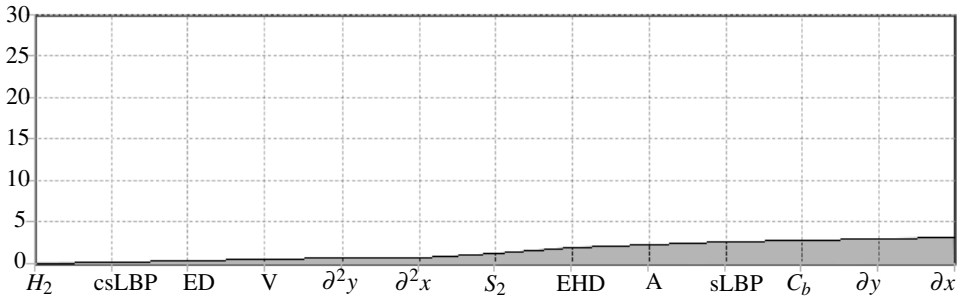
Finally, table 2.3 shows that the mutual information between the texture and colour features. As we have expected it is very low because colour and texture features share little information.

These mutual information values not only are useful for this work but they can also be used in similar works when deciding about which features should be used.

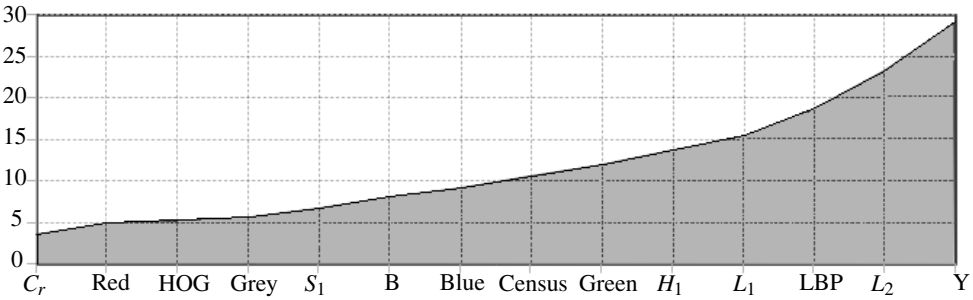
Stage B.2: the feature ranking

In the second stage, mutual information is used to build up a ranking of the features. First, as the mRMR algorithm suggests, the first feature of the ranking will be selected by checking which one seems to be the best to identify the target. In this case, there is barely any prior information about the target since there are not restrictions about his clothes nor the illumination conditions where the robot might be operating. Consequently, we have based this decision on the following assumption: it is very strange to work on scenes where there is no colour at all. Thus, the hue (H_2 in the tables) might be a good descriptor to start with, and therefore this is the first feature in the ranking.

The second feature in the ranking will be the one that, according to the mutual information tables (Tables 2.1, 2.2 and 2.3), has the minimum mutual information with the first feature that has been picked (H_2). The next features will follow the same rule, i.e., the n^{th} feature will always be the one which has the minimum mutual information with the previous $n-1$ features that have already been chosen. Formally, the next feature in the ranking is the feature Y that



(a) First 13th features in the ranking



(b) (Cont.) Next 14th features in our ranking

Figure 2.13: Increasing of mutual information within the features in the visual model, every time that a new feature is added. For space reasons the top graph continues in the bottom one. The x-axis represents the ordered list of features, i.e. H_2 from the top table is the first feature in our ranking, while Y is the last one. The y axis shows the amount of mutual information between each feature on the x-axis, and the rest of features that have already been selected (features on the left).The mutual information increases each time that a new feature is added because there is always at least an small redundancy with the previous ones.

minimises its mutual information with those already in the ranking R, according to Eq. 2.12:

$$MI(Y,R) = \sum_{f \in R} MI(Y,f) \tag{2.12}$$

Finally, the feature ranking that we have obtained can be seen in Fig. 2.13.

Stage B.3: selecting the number of features

The final stage consists in selecting the number of visual features for the human discrimination task, as a trade-off between discrimination power and computational load.

First, in order to measure the performance of each set of features, we use the f-score[54], and in the case that the performance is similar with two models the criterium that will prevail is that the lower the number of features, the better for the robot behaviour.

The f-score is a measure of the performance based on precision and recall:

$$F = 2\frac{precision * recall}{precision + recall} \tag{2.13}$$

$$precision = \frac{TP}{TP + FP} \tag{2.14}$$

$$recall = \frac{TP}{TP + FN} \tag{2.15}$$

where TP stands for *true positives* (i.e. the target's [3] torso is correctly identified), FP stands for *false positives* (i.e. a distractor's [4] torso is identified as belonging to the target), and FN represents the number of *false negatives* (i.e. a target's torso is identified as a distractor). Precision and recall are two measures widely used in machine learning but not suitable for tasks where true negatives (TN) are important. Precisely, in this case true negatives are not relevant, thus f-score is a good way of measuring the performance of the human discrimination task. This is best explained through the following example: imagine that the tour-guide robot is following an instructor across different areas of a building during several minutes, trying to learn routes that it should repeat and show to other people afterwards. If during the demonstration the robot does not detect a distractor (TN), but it keeps detecting the target (TP), it will not affect to its behaviour, hence it will yet be able to accomplish the task and successfully record the route.

Firstly, in order to obtain the f-score values for each visual model we need a test bed. For this reason, three sequences were captured from the robot's camera at the Electronics and Computer Science Department of the University of Santiago de Compostela (Spain). In these sequences, the robot was being controlled by a human in order to follow a target with several distractors walking near the target. It is important to point out the particularly challenging lighting conditions of the corridors at the department. In this regard, Fig 2.14 shows a map of the *light intensities* in the corridors where the sequences were recorded. This map shows how the light intensity ranges from 20lx (dark areas) to over 400lx (right under the lights) in few meters. Moreover, Fig. 2.15 shows some examples of shadows and reflections caused by the light-intensity variations.

[3] The target is the human that our tour-guide robot is following

[4] The distractors are people present in the environment where the robot operates, distinct from the robot's target

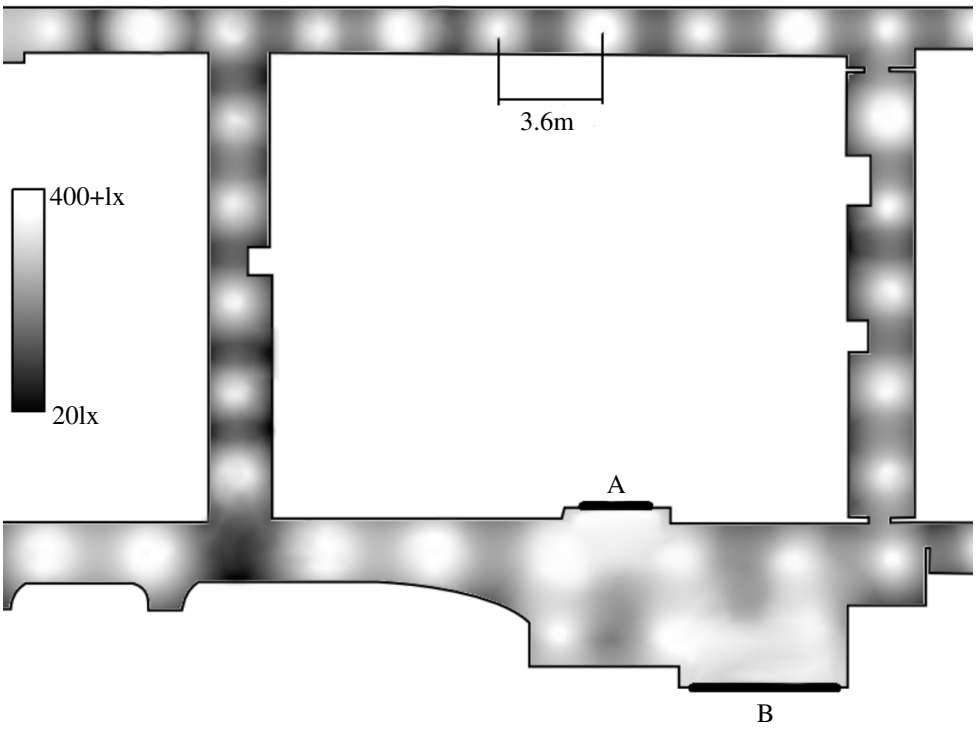


Figure 2.14: Light intensity map of the corridors where the sequences that form the test bed have been recorded. There are numerous fluorescent grille lights whose light beam is very narrow and intense, and which creates bright and dark areas every 3.6m. Moreover, the intensity of some of the lights is half the intensity of the others. There are two sources of natural light: a big window located at point A, and glass door 5 meters width, at point B

In two of these sequences there were two or three people (distractors) interfering with the trajectory of the robot or walking near the target being followed creating partial and total occlusions. In the third one there were over 20 students on the corridors. Since this third environment is crowded with people, the probability of the robot mistaking the target with the distractors is also higher than in the other two sequences. Besides this third sequence was recorded using a low frame rate (half of the frame rate used with the rest of the videos) at a higher resolution. A low frame rate allows the robot to analyse images with higher resolution, but these images might contain more blur when the robot moves fast. Table 2.4 shows some additional statistics about the sequences recorded for this experiment.

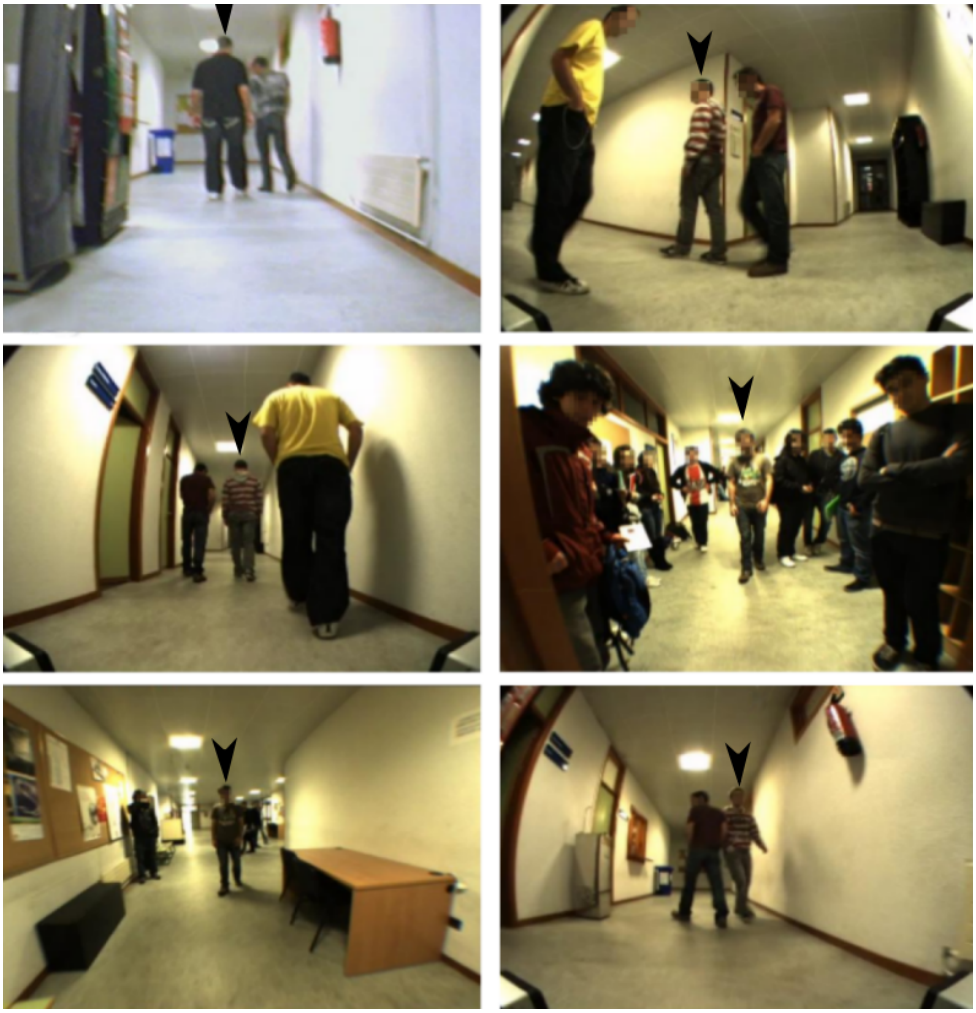


Figure 2.15: Samples frames from the three sequences which have been used as test bed to measure the human discrimination performance in our system. It can be seen how lights create shadows and alter the colour and texture perception, as well as, how crowded the corridors can be at certain moments. The robot's target is marked with a black arrow in the images.

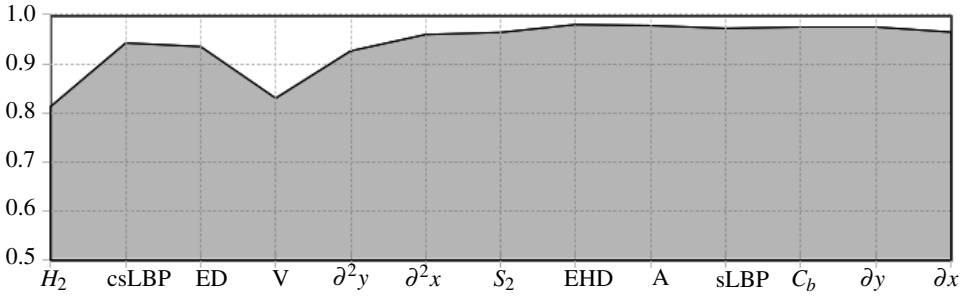
Sequence	Distance	Frames	Torsos/frame
1	40m	887	1.03
2	40m	914	0.81
3	60m	447	1.47

Table 2.4: Statistics from the three sequences that compose the test bed. From left to right: distance travelled by the robot during recording of the sequences, number of frames, average number of detected humans per frame.

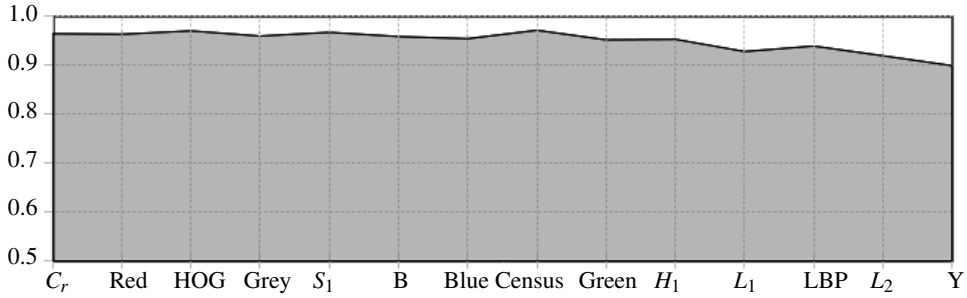
To analyse the performance of the different human models on the three sequences, we need a ground truth to compare with. Therefore, we first ran the human detector algorithm on the three videos to detect people in each frame, and then we manually labelled each person as target or distractor. We then used these annotations as the ground truth to asses the performance of our method in following a target personn.

In order to compare performances and thus find the best size for our feature set, we have first computed the mean f-score in the three sequences using a visual model of the human with only one feature (H_2, the first in the ranking). We then repeated this process iteratively adding new features to the model until the 27 features were included. The features were added in the order obtained during the ranking process (section 2.5.1, *The offline feature ranking*). The results obtained are those shown in Fig. 2.16. We can appreciate the high performance of our system, despite of the challenging illumination conditions previously described: the maximum f-score is 0.987 (a value of 1 corresponds to a situation in which there are no errors). The maximum score was obtained when the visual model was composed of eight features. There is another important aspect quite noticeable in Fig. 2.16, the inclusion of the V feature might seem to reduce the performance of the system, therefore we have repeated these tests without that feature. The maximun f-score obtained in this second test is lower than in the previous one, therefore we can conclude that this feature does not reduce the performance of our system. In particular removing the V-feature from the visual model composed of the first 8 features causes the f-score to be reduced to 0.937, which is less than the 0.987 that we have obtained with the original 8 features. This particular role of the V-feature will be clear when we introduce specific weights to each feature in Sec. 2.5.2, *online feature weighting*.

According to the results presented in this study, and considering the fact that the robot has strong real-time restrictions and hence it requires a visual descriptor with low computational cost, we decided to use the first 8 features of the ranking: the HSV colour space, the second derivatives in both image axis ($\partial^2 x$,$\partial^2 y$), the edge density image based on Canny (ED), the



(a) Evolution of the f-score as we include the first 13th features (x-axis) in the visual model



(b) (Cont.) F-score evolution as we include the last 14th features (x-axis) in the visual model

Figure 2.16: Evolution of the f-score as we increase the number of features in the visual model of the human. The y-axis shows, for each point in the graph, the f-score obtained when the model is built using the set of all features in the x-axis on the left of the point which is being analysed (i.e. the f-score shown for the V-feature corresponds to the f-score when a set of four features is used: H2, CSLBP, Canny and V). The best performance is obtained when the first 8th features are used. For space reasons, the top graph continues in the bottom one.

centre-symmetric LBP (csLBP) and the edge histogram based on the MPEG-7 edge histogram (EHD). This subset of visual features, as well as, the position of the human, will be used from now on to identify and distinguish the robot's target from the distractors.

2.5.2 Online feature weighting

We have just described how we have selected a suitable set of visual features to distinguish the robot's target from the distractors. These features describe the visual appearance of a human, and most of the time all of them are useful to discriminate between humans. However, there might be some situations in which some features are more discriminating than others,

or even situations where some features will not allow that discrimination. As the variability in illumination conditions and people clothes is high, we have designed a method with the necessary adaptability to specific contexts.

The online feature weighting process endows the person following behaviour with the capability to adapt itself to changes such as those generated in clothe's appearances by changes in illumination conditions, or the coming in and coming out of different distractors of the scene. The goal of this algorithm is to dynamically weight the visual features that compose the visual model to enhance the discrimination between the target and the distractors. A similar process has also been studied in the area of image retrieval with query relevance feedback, and consists of measuring the discrimination ability of each feature when differentiating amongst two classes.

In robotics, we can define online feature weighting for human discrimination as the process of dynamically assigning high weights to those features that show a high discrimination ability between the target and the distractors. This is very useful when target and distractors share a similar distribution on some features but differ on the others. We can think of, for example, a target and several distractors wearing similar colour clothes but with different patterns. In this case it would be more useful to focus the similarity measure (Eq. 2.4) on the texture while discarding the colour features. For this, we have replaced Eq. 2.4 with a new measurement that considers the weights w_f of the different features:

$$similarity = \sum_{f=1}^{n} w_f d_f \quad \in [0,1] \tag{2.16}$$

Initially, the weights are the same for all n features, i.e., $w_f = \frac{1}{n}, \forall f = 1, ..., n$, but as the robot proceeds moving in the environment, the weights will be updated according to the online feature weighting process that we propose.

Fig. 2.17 shows an an schematic representation of the algorithm and its role in the whole human discrimination process. Essentially, we can observe that using information from the target and the distractors we can compute a score for each feature. This score is in charge of determining how discriminant is a feature, and we use it to update the weights of each feature, according to:

$$w_f = w_f + score_f \tag{2.17}$$

Given that the value of the scores is not bounded, after applying this equation, it is important to normalise the weights so that their sum is one. We restrict the weight values to the positive

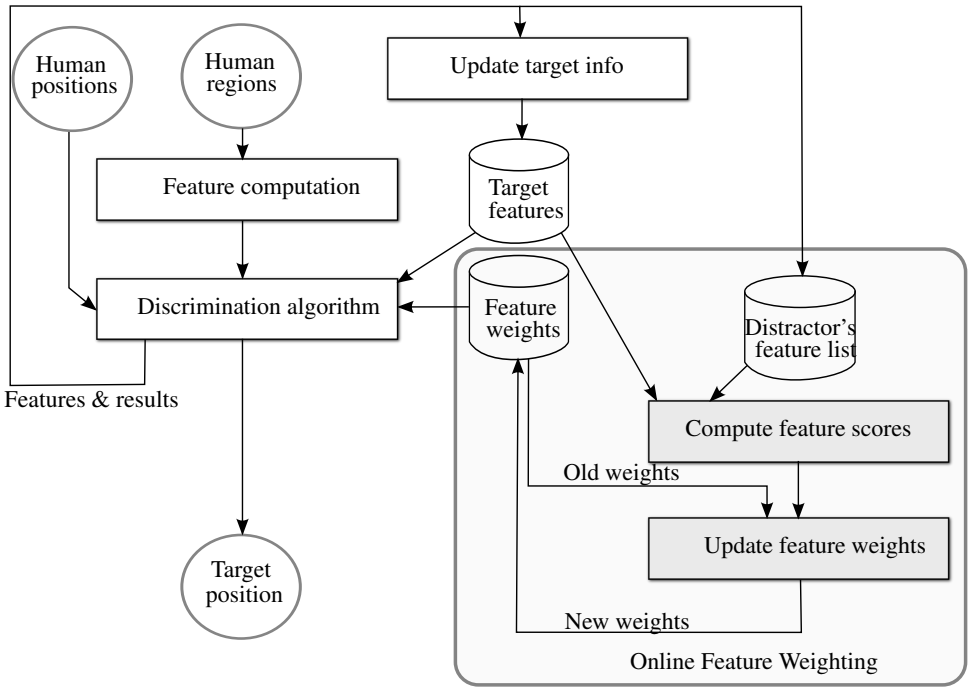


Figure 2.17: Scheme of the human discrimination process with the online feature weighting process shaded in light gray.

rangen this way the lowest possible weight is zero (when the feature is temporary ignored), while the highest is one.

It is obvious that how we compute the score for each feature will determine the usefulness of our online feature weighting process. For this reason, we had proposed our own scoring function [18], but since there were other alternatives that have been used for similar purposes in the past in the area of image retrieval, we have conducted a study to test which one suits best our task [19], as well as to compare our solution with the case when the features are not weighted.

We have compared our own scoring function with the ones described in [55], which from a representative set of the existing ones. In the following description of candidate scoring functions, n_1 represents the class of the target, while n_2 represents the class containing the distractors, while $\mu_{f,1}$ and $\mu_{f,2}$ are the mean values of the f^{th} feature in n_1 and n_2 classes, and $\sigma_{f,1}$ and $\sigma_{f,2}$ are the standard deviations of the f^{th} feature in n_1 and n_2, respectively.

– *Inverse-Sigma.* This function does not use information about the samples included in n_2 and prioritises those features with low standard deviation on the target (n_1 class).

$$score_f = \frac{1}{\sigma_{f,1}} \tag{2.18}$$

This function requires that the features for the n_1 class follow a unimodal distribution.

– *Delta-Mean.* This function determines the importance of each feature measuring the difference between the mean value for feature f in the class n_1 and the mean value for the same feature in the class n_2, these differences are normalised by the sum of the standard deviations:

$$score_f = \frac{|\mu_{f,1} - \mu_{f,2}|}{\sigma_{f,1} + \sigma_{f,2}} \tag{2.19}$$

This scoring function requires the distributions for each feature and both classes to be unimodal.

– *Membership.* The idea behind this scoring function is to measure how separable the n_1 and n_2 classes are, by building several classification matrices. The classification rule is simple, first we establish the range $(\mu_{f,1} - k * \sigma_{f,1}\ ,\ \mu_{f,1} + k * \sigma_{f,1})$, where k vary from 0.5 to 4. Then, if an element belongs to n_1, then it must be within the mentioned range, and if a element belongs to n_2, then it must be outside that range. After building the matrices, one for each k, we can compute the score as the maximum ratio between the samples that are correctly classified and all the samples:

$$score_f = max(\frac{a_{11} + a_{22}}{a_{11} + a_{12} + a_{21} + a_{22}}), \forall a_k \tag{2.20}$$

where $a_{x,y}$ are the x, y cells of the confusion matrix a_k. Once again, this score requires every feature distribution in n_1 to be unimodal.

– *Entropy of target.* This score measures, for every feature, how clustered the samples in n_1 are, the more the samples are grouped together the smaller the entropy of the feature will be:

$$score_f = log(b) + \sum_{i=1}^{i=b} (p_i log(p_i)) \tag{2.21}$$

where b is the number of bins used to build the histogram for feature f and p_i is the probability for bin i. This score does not make any assumption about n_1 but it does not consider n_2.

– *Entropy of the target and the distractors*. This score includes the entropy of the features that describe the distractors in Eq. 2.21. The main problem of these entropy-based functions in the context of our work, is that a feature can get a high score only because it shows a non-random behaviour in n_1 and n_2, regardless of its ability to discriminate.

$$score_f = 2log(b) + \sum_{i=1}^{i=b}(p_i log p_i + q_i log q_i) \tag{2.22}$$

where q_i is the probability of getting a value of feature f in n_2 (distractors), included in bin i.

– *Kullback-Leibler asymmetric distance*. We can use this measure to determine how much the distribution of a feature in n_1 differs from the distribution of the same feature in n_2, i.e., how correlated both distributions are:

$$score_f = \sum_{i=1}^{i=b}(p_i log \frac{p_i}{q_i}) \tag{2.23}$$

– *Kullback-Leibler symmetric distance*. This scoring function adds a new term to Eq. 2.23, in order to become symmetrical:

$$score_f = \sum_{i=1}^{i=b}(p_i log \frac{p_i}{q_i} + q_i log \frac{q_i}{p_i}) \tag{2.24}$$

– Difference of histogram distances. This is the scoring function that we have proposed [18], and which is based on the Bhattacharyya distance shown in Eq. 2.5. The idea behind this score is that the best feature should be the one that minimises the Bhattacharyya distance between the feature histogram of the region classified as target and the one in the target model ($d_{f,1}$, in Eq. 2.25) while it also maximises the average distance between the histogram of the same feature in the distractors and that of the current target model ($\bar{d}_{f,2}$ in Eq. 2.25):

$$score_f = \bar{d}_{f,2} - d_{f,1} \tag{2.25}$$

In order to select which scoring function suits best our problem, we have used the same three video sequences described in section 2.5.1. First, we ran the human discrimination algorithm without online feature weighting (Eq. 2.4), in order to obtain the number of times

Score function	Robot's mistakes in...		
	seq. 1	seq. 2	seq. 3
Bhattacharyya	2	**1**	8
KL-Sym	**1**	**1**	**6**
KL-Asym	6	9	**6**
Membership	11	6	22
Entropy n1n2	10	2	16
Entropy n1	44	2	24
Delta-Mean	48	**1**	29
InvSigma	175	4	13
Baseline	14	6	18

Table 2.5: Number of times that the robot confuses a distractor with the target (robot's mistakes) in each one of the sequences in our test bed, when different scoring functions are used to determine the importance of each feature. The *baseline* corresponds to the case where we all the features have the same importance, i.e., the same weight.

that the robot confuses a distractor with the target. We obtain this number by comparing the results of the human discrimination algorithm with the ground truth obtained for each video. This first result represents the 'baseline' (last row in table 2.5) which will allow us to measure the improvement achieved when the scoring functions are used to weight the features.

Therefore, after obtaining the baseline, we have run the same experiments but weighting the features (Eq. 2.16) using each of the candidate scoring functions. Table 2.5 shows the number of times that the robot confuses a distractor for the target, when each one of the scoring functions is used to process each one of the three videos. It can be seen that our scoring function (the difference of histogram distances), and Kullback-Leibler symmetric scoring functions are able to significantly reduce the confusion of a distractor for the target, improving the results obtained when not weighting the features.

These results allow us to confirm that weighting the feature set with complex functions (such as those which use the distributions of both the target and the distractors) improves the performance of the human discrimination in our tour-guide robot.

Additionally, we want to show some graphical examples of the results of this algorithm and the importance of weighting the features. In this regard Figs. 2.18 to 2.20 show specific situations where the robot does not confuse the target with a distractor thanks to the online weighting of the features.

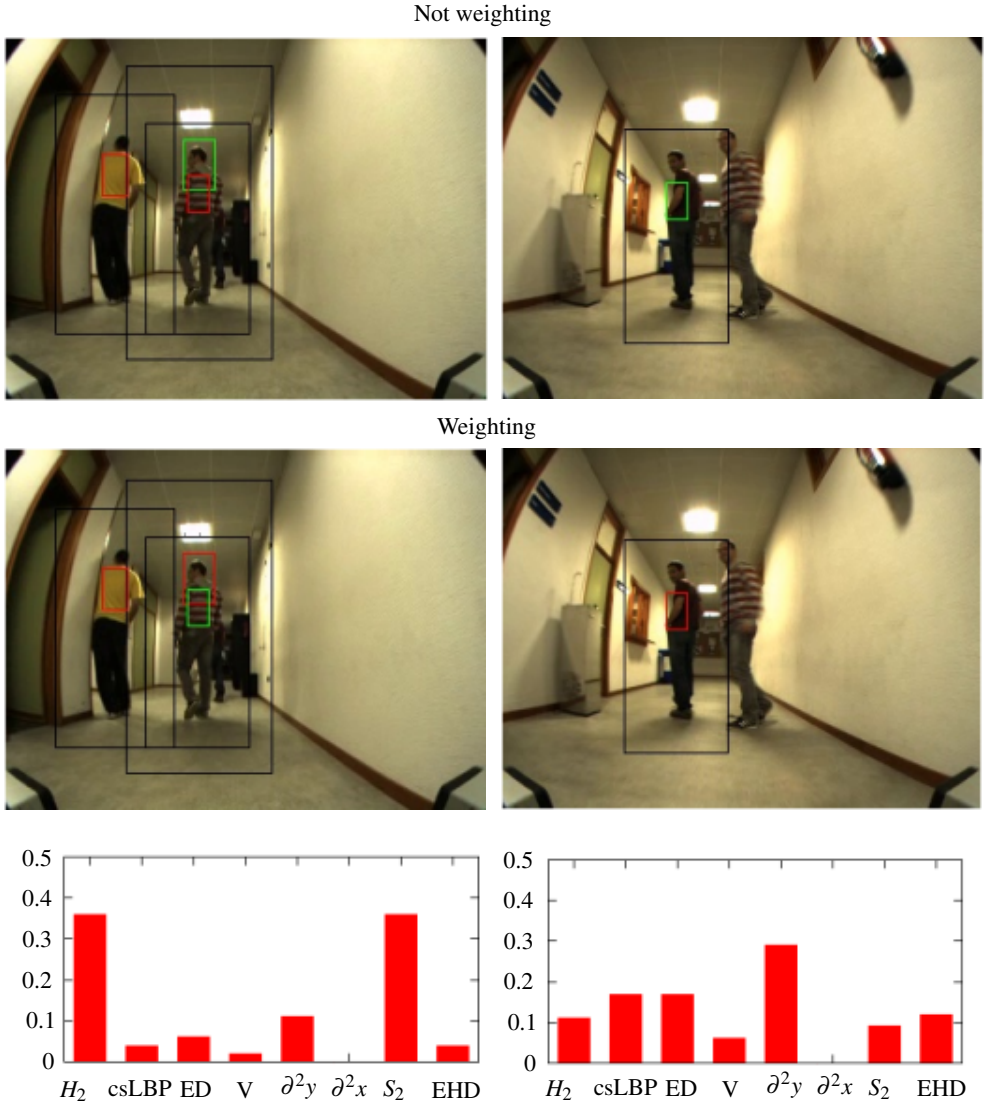


Figure 2.18: Two situations where the online feature weighting selects the most appropriate features to discriminate the target (brownish with horizontal stripes) from the distractor present at each moment. Left column: the distractor's torso is of a different colour, therefore H_2 and S_2 are selected. Right column: the distractor's torso colour (when mixed with the background) is similar to the target colour, however the horizontal stripes are a good choice to discriminate amongst both torsos, therefore the weight of $\partial^2 y$ is increased.

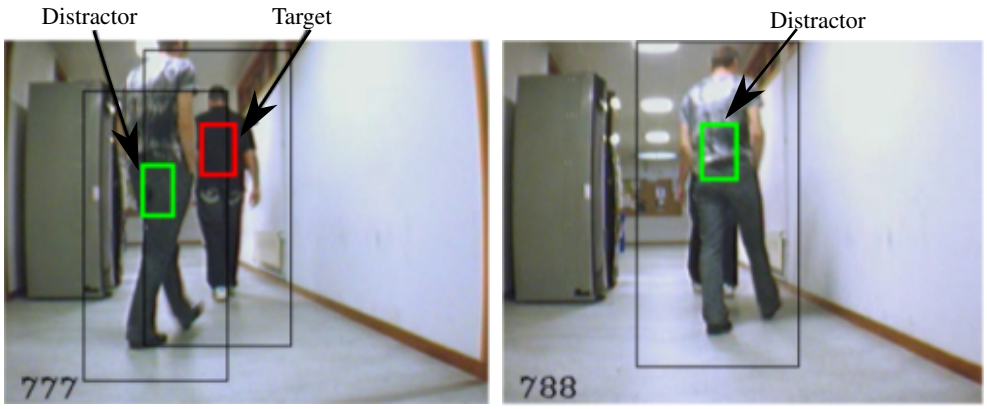


Figure 2.19: Two frames captured from a video sequence where the robot is trying to follow the human dressed in black. The human discrimination without online feature weighting fails to distinguish the target from a distractor due to very similar clothing. Fig. 2.20 shows how we solved this problem by using online feature weighting.

In particular, Fig. 2.18 shows how the relevance of the features depends on which target is being followed and which distractors are present at each moment. In the left column of this figure we can see how the relevance of the colour features (hue-saturation) is the highest when it comes to discriminate between a yellow and a brownish torso. Nevertheless, some seconds later, in the right column, we can see how the most relevant feature is the vertical derivative ($\partial^2 y$), which is used to discriminate between a stripped torso and a flat torso with similar colour (both are brownish). Fig. 2.19 and Fig. 2.20 illustrates a similar situation, where three features are selected over the rest in order to discriminate between two torsos with similar colour distributions.

Finally, we have carried out some experiments to reinforce our conclusions from the online feature weighting process. More specifically, in this experiment we wanted to check whether the weight of some feature was marginal and could be excluded from the set, thus reducing the computational burden of the robot. In this regard, Fig. 2.21 shows the average weight of all features after processing the three sequences in our test bed. It is important to notice that in both graphs the HSV colour space represents about 60% of the total feature weights while, on the contrary, the importance of the texture features csLBP and EHD are reduced to about 15% of the total sum of the feature weights. The main difference amongst the weights of the

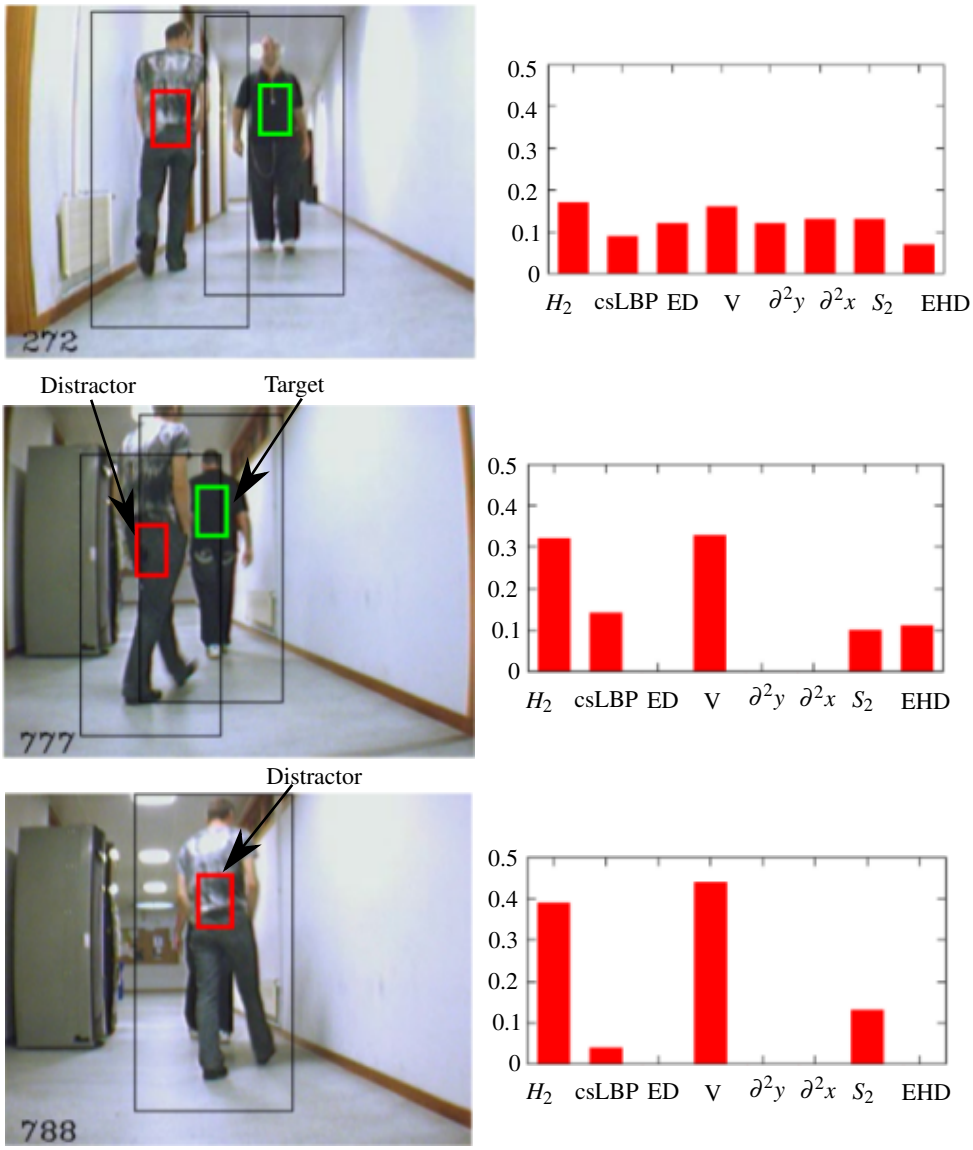


Figure 2.20: Illustration of how the online feature weighting solves the problem of Fig. 2.19. Top row shows the frame where a distractor is first detected and the weights start to change but are still almost equal between all the features. Middle and bottom row show the same frames of Fig. 2.19, where the humans have now been correctly identified, thanks to selecting the most discriminant features.

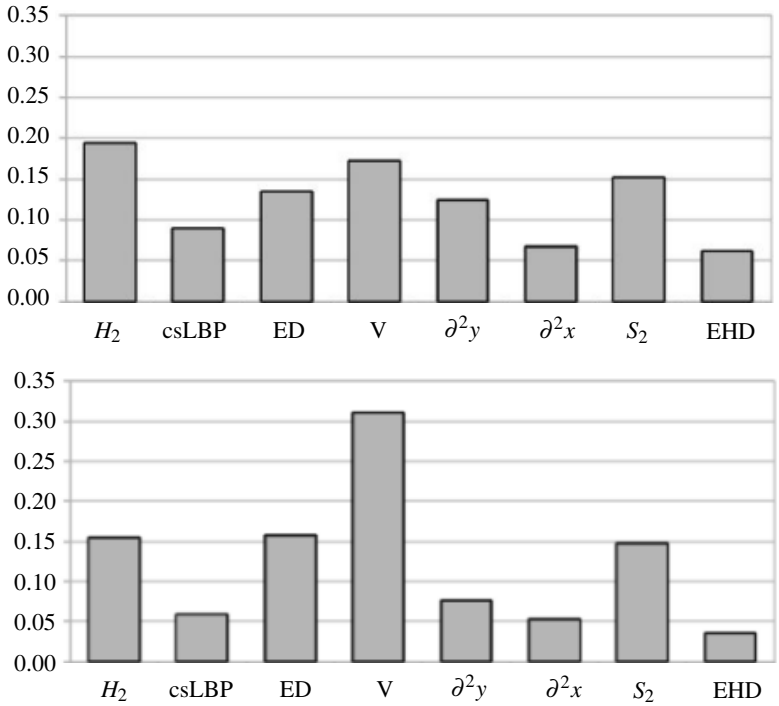


Figure 2.21: Average feature weights obtained when the *Difference of histogram distances* scoring function (top) and *Kullback-Leibler symmetric* scoring function (bottom) are used.

features when each scoring function is used lies in the importance of the V feature (from the HSV colour space).

The previous figures allow us to conclude that all features have a non-zero weight and therefore all of them contribute to identify the target. Nevertheless, the average weight is not a good measurement to determine whether one feature is more relevant than the others. To prove this, it is enough to consider Fig. 2.22. This new figure shows the evolution of the weights for three features (Hue, Value and MPEG also referred as EHD). Despite of the fact that the EHD seems to be irrelevant according to Fig. 2.21, there are occasions where the weight of the EHD is higher than the weights of H and V features (around frames 300 and 630 in Fig. 2.22). Therefore, it is straightforward to conclude that a feature can be irrelevant most of the time but there might be particular occasions where that same feature is crucial to

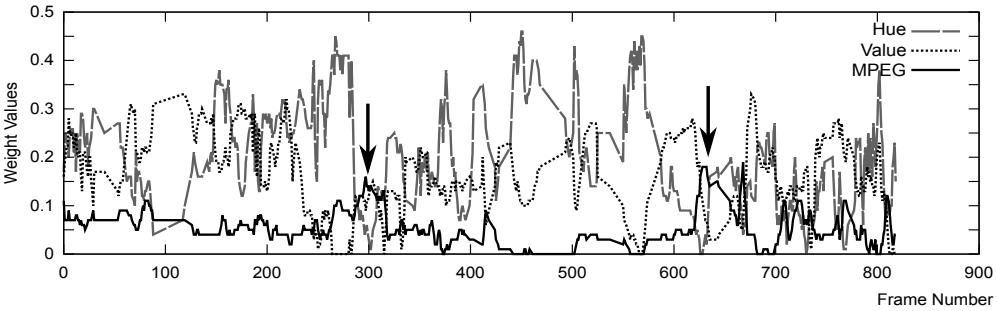


Figure 2.22: Evolution of the Hue, Value and EHD features weights during the first sequence. The arrows in the figure highlight those frames in which the EHD feature gets a higher weight than hue and and value features.

avoid a mistake. In this sense, this algorithm plays an important role on dynamically selecting the most relevant ones at a time.

2.6 Testing the person-following behaviour in several demonstrations

During the time that we have been developing our tour-guide robot, we have taken it outside of our department to perform several live demonstrations. More specifically we have been at:

- The secondary school *IES Ramón María Aller Ulloa*, located at Lalin, Pontevedra.
- The elementary school *Colexio público Pio XII*, located at Santiago de Compostela, A Coruña.
- The secondary school *IES Rosalia de Castro*, located at Santiago de Compostela, A Coruña.
- The engineering school *Escola Técnica Superior de Enxenaría*, located at Santiago de Compostela, A Coruña. We have showcased our robots for a whole day in an open event.
- Two consecutive years (2012-2013), during the *DÃa da ciencia en Galego*, at the faculty of sciences, located at Lugo. We have showcased our robots to four organised groups.

– During two years (2013-2014) we have been part of a science day at the faculty of sciences at Lugo.

– Two consecutive years at the Domus Museum, located at the city of A Coruña. We have showcased our robots, first in an open event, and then three organised groups of people during a whole day.

Also, since the second half of 2012, our tour-guide robot has been part of the organised visits to our research centre. In these visits, many secondary schools, as well as many associations and heterogeneous groups in general, could test the person-following behaviour, as well as the other abilities of our robots. Probably more than 500 people have tested our tour-guide robot since we started to showcase it.

We want to specially comment on the Domus museum, which is located at the city of A Coruña, Spain. This museum contains many light sources with different colours and intensities and the floor is very reflective (an example of floor reflections can be seen in Fig. 2.24). Despite of this, the robot was able to follow a human through the environment. Only occasionally the target had to wait for the robot, but this was due to the fact that the robot moves slowly in narrow areas or narrow passages for safety reasons. The speed is limited to 0.6 m/s, and this value is only achieved when the target is far from the robot. In this regard, Fig. 2.25 shows the speed of the robot according to the distance from the robot to the target during this test. In this figure we can notice how the distribution of points fit two lines; the top line represents the normal speed of the robot when only the distance to the target matters, the bottom line is the speed when the robot operates in *safe mode* due to narrow areas or when someone is passing nearby, in this case the speed is determined not only according to the distance of the target but also considering the obstacles in the environment.

Finally, we have recorded the target position relative to the robot during three different demonstrations (Fig. 2.26). We can observe how the robot is able to maintain the forward direction towards the position of the target, on the other hand we can also see that the distances amongst the robot and the target range from just few centimetres up to 5.5 meters.

2.7 Conclusions and future work

Real-time human detection, recognition and tracking are relevant topics in today's robotics. Particularly, these abilities will allow personal robots to perform complex tasks which involve a high level of human-robot interaction. This chapter describes two novel proposals: a) an

Figure 2.23: Pictures of several demonstrations of our robot during different events.

Figure 2.24: Frame captured from the demonstrations at the Domus museum (A Coruña) when the robot was following the person in the centre of the image. In this frame it is possible to observe how several light sources are causing many reflections.

efficient human detection algorithm for robots with almost no pose restrictions or the requirement of additional gadgets; and b) a human recognition and tracking algorithm for a robot that requires detecting and following a specific human (target), in crowded environments where the illumination conditions might change drastically.

First, we have have proposed an algorithm for fast and robust human detection which uses depth information. More specifically, it searches for human-like blobs in depth images from a range camera and combines those results with those of a leg detection algorithm performing in the range data from a laser scanner. The main advantage of this algorithm is that it does not rely on human features that are only visible at certain poses like the face or the head-shoulders sillhouette. Therefore, the algorithm is able to detect a human regardless of his pose. On the contrary, the main drawback is that it is not currently able to segments to humans that are touching each other, i.e. two humans next to each other without space between their shoulders or arms. This limitation is clearly identified and we propose that it should be addressed in

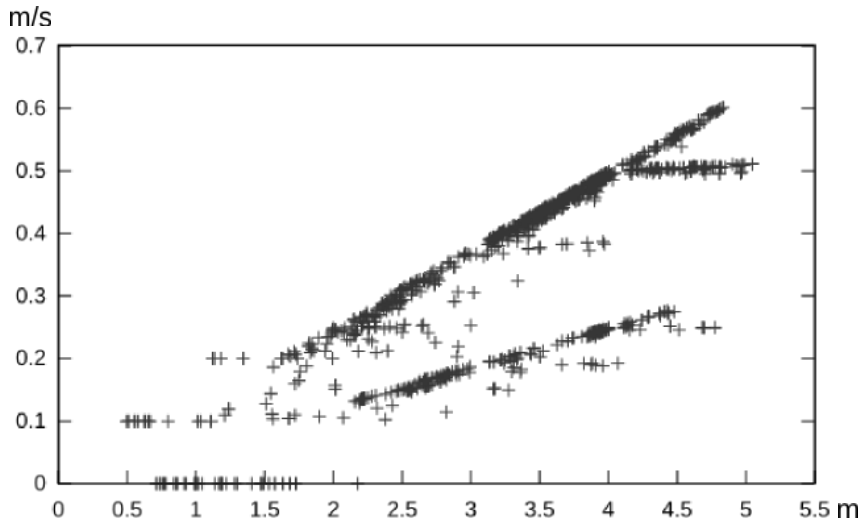


Figure 2.25: Speed of the robot according to the distance to the target. The y axis represents the robot's linear speed (m/s), while the x axis represents the distance (m) from the target to the robot. This data was logged while the robot was performing the person following task.

future works. More specifically, the legs or even the human heads can help to segment the human blob into the actual two humans.

Second, we have presented a mechanism that combines colour and texture features to characterise human appearance. Due to the great number of colour features and texture descriptors that have been already published in the past; we decided to carry out a deep analysis of a selection of the 27 most common. In particular, we have analysed this selection of features to determine to what extent some of them provide information that is redundant, and therefore, can be discarded. The analysis shown in this paper might be very useful for future works that decide to use colour and texture for the same or similar tasks. Moreover, this algorithm also includes a high level of adaptability to the characteristics of the environment. The core of this adaptability belongs to a scoring function, which is able to determine in real-time the best features to distinguish the target from the distractors. Once again, we did not want to restrict our work to only suggesting our own scoring function, instead we did an in-depth analysis of state-of-the-art scoring functions. From this analysis, we can conclude that for the human recognition task it is important to consider not only information regarding the target

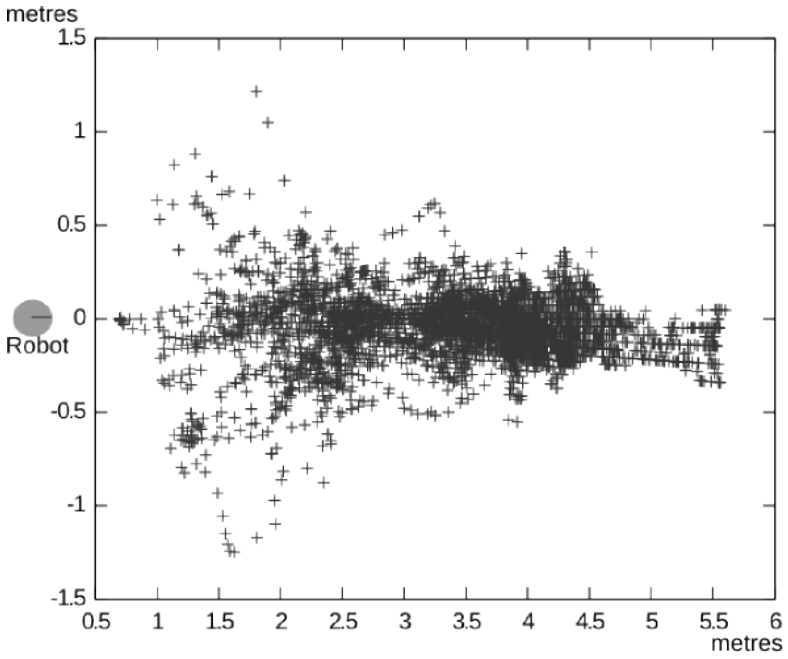


Figure 2.26: Relative position of the target with respect to the robot (which is located at the coordinates 0,0).

but also some relevant aspects related with the distractors. This is the reason why the scoring functions which use the distributions of the features corresponding to both, the target and the distractors, obtain the best results.

Nevertheless, there are some particular situations not addressed in the current human recognition algorithm. One is related to when a human has different features depending on the angle from which the robot sees him. An example is when a human has an open black jacket with a white t-shirt only visible from his front side. The current implementation of our algorithm can overcome this problem only if the person turns slowly, and the model gets updated to switch from the black jacket to the white t-shirt. A better solution than this one would be to maintain several parallel models of the human that is being tracked. However, that proposal also opens other problems that should be addressed, i.e. when to switch from one model to the other or when to update each one. Summarising, those are open problems that should be solved in an evolution of this work.

Finally, we have integrated these algorithms to build the person-following behaviour in our tour-guide robot. The behaviour was tested in real environments such as the Domus Museum (located at the city of A Coruña), or elementary and secondary schools with people of all ages.

CHAPTER 3

HUMAN ROBOT-INTERACTION

Any service robot should possess some kind of human-robot interaction (HRI) to be able to communicate with its users. This is even more clear when speaking about personal robots which work closely with humans such as our tour-guide robot. In the previous chapter we have already described an essential part of the human-robot interaction: how to detect, identify, discriminate and track humans. Even tough those are also part of the robot's interaction capabilities, we separated them from this chapter to ease the reading.

There are two types of users of our robot: the route instructor, and the visitor of exhibitions. The robot's instructor will be able to teach new routes to our robot, as well as, to record voice messages which describe points of interest along the route. Regarding the visitor of an event, he will have the opportunity to select a route from those that were taught by the instructor. Once, a visitor has selected a route, our robot will follow the route path, stopping at those points of interest where the instructor has recorded a voice message, in order to reproduce it.

Similarly, we can classify the Human-robot interaction of our tour-guide robot depending on the direction of the communication:

– On the one hand, there is the communication from a human (transmitter) to the robot (receiver), which consists in commands to the robot. The instructor will need to: a) tell the robot that he will be teaching a route; b) tell the robot when to start and stop recording a voice message within the route; c) tell the robot that the route teaching process has finished. The visitor of an event will need to: d) ask the robot about the available routes; and e) command the robot to show him the desired route.

– On the other hand, there is the communication from the robot (transmitter) to the human(receiver). In our case, we differentiate between the information given with the intention of easing the interaction (a graphical user interface with augmented reality and voice feedback), and the information given when reproducing a route (voice messages and sounds). Thus, our main challenges are: a) to give enough feedback to the user to allow him to understand the robot's actions; and b) to provide the user with information about the route that the robot is demonstrating.

In this chapter we describe the interaction scheme that we have designed for our tour-guide robot. It allows the users to send commands to the robot using hand movements. Moreover, those users continuously receive feedback from the robot: what are the available commands at a given moment, what is the robot doing, why has the robot stopped. In order to send this feedback, we introduce an augmented reality graphical user interface (AR-GUI) with voice feedback that will guide the user through the necessary steps to interact with the robot. Our experience in several live robot demonstrations taught us that the user of a robot is usually willing to cooperate with it, in the case that he knows how to help it. Precisely, this is the point where we aim to contribute the most to human-robot interaction: easing the cooperation between robots and humans by making them aware of what the robot is expecting from them.

3.1 A description of the challenges

In order to allow the users of our robot to communicate with it using hand movements there are several challenges that we have to address:

– First, hands need to be detected. Similarly to detecting humans, it is a complex task because it is a non-rigid object which can adopt a wide variety of poses and might be subject to frequent occlusions. In this regard, once we have detected the hands, we need to keep track of them.

– Second, a set of hands gestures must be recognized. We need to recognise different temporal patterns with significant variations amongst different users of the robot.

– Finally, an augmented reality graphical user interface with voice feedback would make the communication easier. We need to find intuitive virtual elements and short but descriptive voice messages that help the users of our robot understand what they need

to do. Moreover, we need to display them at the right time, and avoid overwhelming the user with too much information or non-relevant information.

3.2 State of the art

In section 1.2 we have already introduced that many previous tour-guide robots suffered from limited human-robot interaction capabilities. There, we have also reviewed the current trends, improvements and efforts to solve this problem during the latest years. In this section we open the review to any which has dealt with the human-robot interaction problem obtaining a more detailed view of the existing proposals.

Human-Robot Interaction (HRI) is a broad area of research which has been steadily growing for many years now. Many works have shown the importance of improving human-robot interfaces such as: to take care of the elderly [56], to command an assistant robot [57] or to provide route directions such as in [58] and [59]. HRI covers different areas of research since there can be different types of interaction. First, there is the *speech based interaction* [59], which includes both speech recognition and text to speech, which are themselves substantial ongoing research areas. Second, many works use *facial expressions* to express emotions and communicate with the user [60] . Third, we have the *gesture based approaches*, which include both static [61, 57] (hand symbols) and dynamic [62, 63] (hand or arm movements). And last, there is also interaction through *graphical user interfaces* [56]. Nevertheless, most works try to combine various types of interaction imitating the way humans communicate. For example, speech on a robot is usually combined with hand gestures [59], or with face expressions [64] or even with a graphical user interface (GUI) [56]. These combinations seek robustness but also try to provide a more natural and human-friendly type of interaction.

In the specific case of communication with a tour-guide robot, in a first place, touch screens revealed as a cheap, easy and reliable way of selecting destinations for tour-guide robots [65, 66, 67, 68], thus they have become a common solution in any tour-guide robot. However, touch screens can be perceived as a temporary solution while more and more research is being done in natural, human-like interaction types. For example, some researchers have been using speech recognition in recent tour-guide robots, such as in *Urbano* [12], and voice feedback, as in Gockley et al. [41]. Although speech communication is natural for humans, recognising human speech does not yet prove to be a robust solution in crowded places. This limitation is well-known and can be observed in recent works [66, 14], which had to

discard the use of speech recognition in real world environments. Another natural type of interaction that has also been explored but has not yet become popular is hand gestures: the famous ASIMO [69] was able to recognise some hand movements such as waving or hand swings.

Thanks to recent hardware advances, and in order to contribute to natural ways of communication, we have decided to explore the possibilities of a gesture based interaction. We believe that, using the new hardware advances, gesture interaction can be a robust, easy and low-cost solution for human to robot communication. Therefore, we choose to send commands to our robot with hand movements, and by *touching* virtual buttons in our robot's augmented reality graphical user interface (henceforth, AR-GUI). The AR-GUI eases the whole process with both visual and voice feedback. We are not aware of any other previous tour-guide robot that has been equipped with the combination of a hand gesture interaction and a graphical user interface with augmented reality.

Hand movement recognition has been deeply studied over the recent years. Initially, the use of wearable sensors (accelerometers, EMG sensors, gloves, etc) was necessary [70, 71]. Nevertheless, nowadays there are new sensors, such as cameras using structured light to compute depth information (Microsoft Kinect, Asus Xtion), which opens up affordable ways of detecting hand positions and thus recognising gestures. The most widely used approaches for recognising the pattern of hand movements include finite state machines [72, 63], hidden Markov models [73, 74], and Dynamic time warping [75]. In our case, we have evaluated the dynamic time warping method and the design of a finite state machine for hand movement recognition.

3.3 Interaction with our tour-guide robot

The interaction scheme that we have designed is intended to let everyone use our robot with an small training period. That is, anyone visiting an event where the robot operates could demand a route, and what is probably most important: anyone should be able to teach routes to it, with a minimal explanation of how the robot works.

As we have already introduced, the solution that we have designed can be divided in two areas. The first one, the augmented reality graphical user interface (AR-GUI) with voice feedback, centralises the flow of information from the robot to its users. The second one, the hand interaction which is the way that a user has to send commands to our robot.

The centre of the human-robot interaction of our robot is its screen. In this screen we display the AR-GUI, in which the users can obtain information about the robot's state, and about what is the robot expecting from them. In this sense, this AR-GUI displays an image of what the robot sees, alongside virtual elements (i.e. buttons, boxes) or augmented scene (i.e. shading the user's body with a specific colour, erasing objects not visible for the robot). Moreover, the AR-GUI will reproduce voice messages at certain moments of the interaction. For example, if our guide robot is following a person and it might warn that person with the voice message *please, walk slower!'*, allowing that person to realise that the robot might lose sight of him.

Using the AR-GUI the humans can *touch* and *push* a set of virtual buttons by simply putting their hands over them. Using this interface, a visitor is able to select a route from a menu with several buttons, or an instructor can tell the robot when to record a voice message. Moreover, in order to approach a more natural interaction, we have associated some hand movements to certain commands, i.e. a user can *wave off* in order to tell the robot to stop following him.

We have divided the rest of this section in three parts. In the first part, we describe how does our robot detect hands, tracks them and recognise its movements. In the second part, we describe the latest version of our augmented reality graphical user interface with voice feedback. Finally in the last part, we present two studies that we have conducted to measure the quality in use of our tour-guide robot. These studies were used to reveal and correct the weakness on the communication process.

3.3.1 Hand detection, tracking and movement recognition

The two main challenges in hand gesture communication are: the localisation and tracking of the hands and the recognition of the gestures (or hand movements). Figure 3.1 shows the flowchart of our proposal to detect a command from a user. The process starts as soon as a set of humans are detected and finishes when a command has been recognised or when the system fails to detect the hands of the user. Essentially, it can be divided into two stages: the initialisation, in which the robot detects the hands, and the recognition stage, in which the robot tracks the hands and recognises a command using the patterns of the hand movements.

The *initialisation stage* consists in detecting the hands of the robot's user for the first time. An user of our tour-guide robot will have to go through this stage every time that he wants to start a new gesture interaction (i.e: when he is being followed and wants to start recording

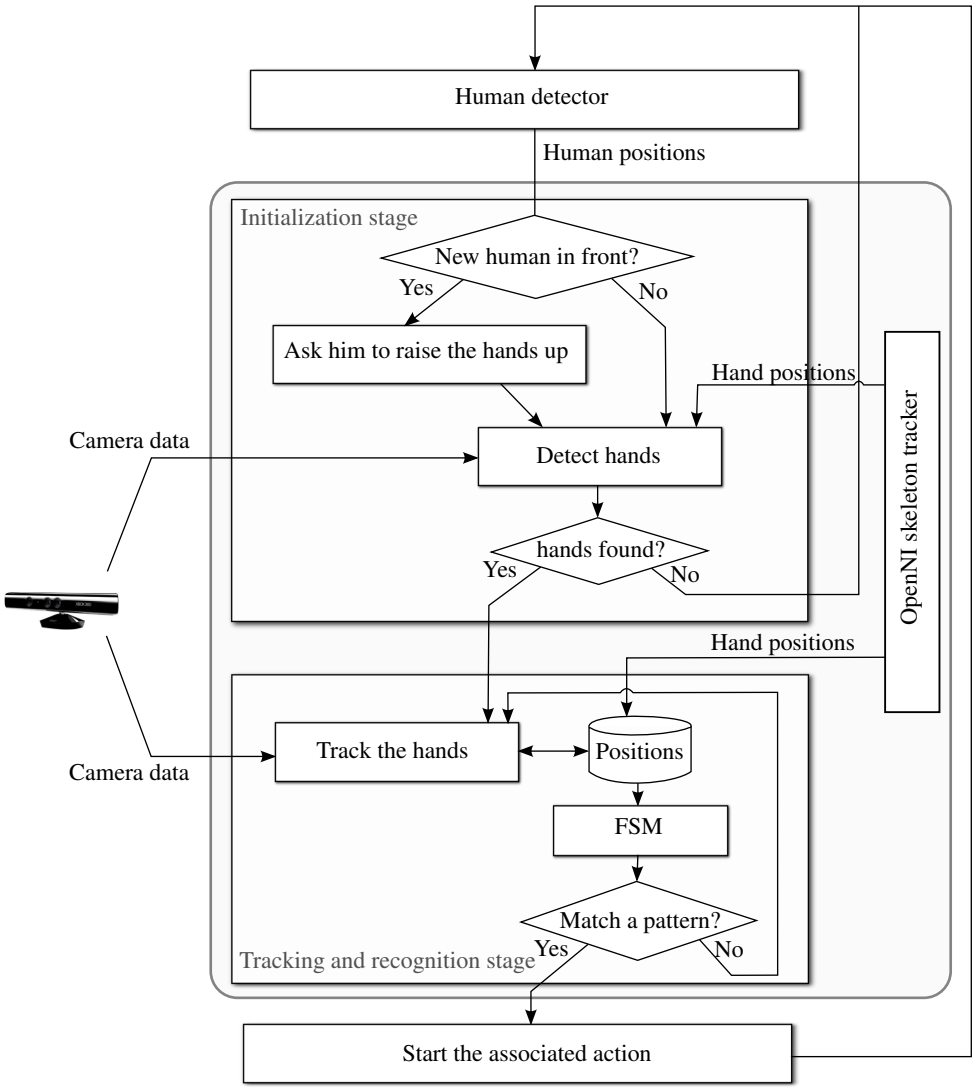


Figure 3.1: Flowchart for the recognition of hand movements. It includes three main processes hand detection, tracking and pattern matching.

Figure 3.2: The robot is detecting a human who stopped in front of it. The human raises his hands up towards the robot. This allows the robot to detect the hands and start tracking them, in order to recognise commands from the human.

a new route, or when the instructor wants to record a voice message in a route). This stage starts as soon as an user stops in front of the robot, and can be performed in two different but complementary ways:

a) Initialisation pose. The user will be asked to hold his hands up towards the robot as illustrated in Fig. 3.2. This pose allows the detection of hands in the following way: depth data provided by range camera is analyzsed, in order to search for two hand-sized objects located in front of the user at a certain height. The height we are using is a percentage of the user's own height. The initialisation pose was in general accepted by the users who tested the system, who did not consider it an awkward requirement.

b) Skeleton tracking. In order to automatically obtain the user's hands, and avoid the use of a initialisation pose, we use a popular solution for skeleton detection and tracking. More specifically, we use the OpenNI skeleton tracker [1], which relies on background subtractions to detect humans and extract their skeleton. Therefore, although this way of obtaining the hands avoids the use of the initialisation pose, it does not work when the robot is moving.

In case that neither methods are able to detect the hands, the process finishes and will be restarted when new humans are detected, which usually happens in about 30ms. This allows that, when a user steps in front of the robot, it will try to detect his hands at least 20 times per second. However, if either of our proposals is able to detect the hands, we start the recognition stage.

The *recognition stage* consists of tracking the hand positions to match the patterns of the gestures. In order to track the hands (Fig. 3.1), we search for hand-sized objects near the previous hand position. When the hand is found, its new position is stored with the rest of the positions to build its trajectory. Since we use a naive method to track the hand, it is possible that the tracker confuses other objects or parts of the human body with the hand. Therefore, when available, we use the hand positions computed by the skeleton tracker to correct our tracker.

We can define the trajectory for each hand as $\vec{P} = \{p_0, p_1, ..., p_n\}$, where p_0 is the most recent position and p_n is the oldest position stored. We currently store the latest 50 positions ($n = 50$, which is equivalent to about two seconds of data). As we have described in this chapter, we have evaluated two options for gesture recognition the possibilities of detecting those hand movements using the popular dynamic time warping (DTW) method, and a finite state machine (FSM) for each gesture.

Dynamic time warping

The dynamic time warping is a popular algorithm for measuring similarity between two temporal sequences which may vary in time or speed. Precisely, hand movements can vary significantly from person to person. For this reason, DTW is widely used in gesture recognition. It requires a model of each gesture that we want to recognise. This model can be manually created or automatically extracted from a set of examples.

[1] $http://wiki.ros.org/openni\_tracker$

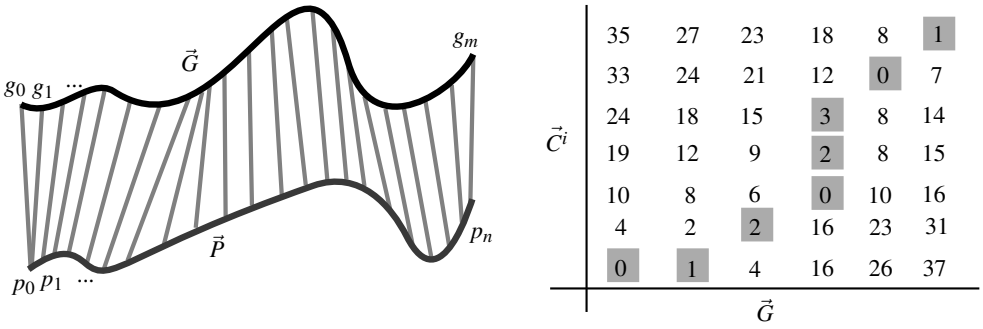


Figure 3.3: Dynamic time warping matching illustration (left) and warping matrix (right).

We have created a model for each gesture as the average of a set of example hand trajectories described by different people while executing the gesture. We can define the model of each gesture as $\vec{G} = \{g_0, g_1, ..., g_m\}$, where m is the number of points in the model. Then, in order to compare $\vec{G}$ with the set of positions of our hand $\vec{P}$, we proceed as follows:

1. For each $iin\{5..n\}$, we create the subvector $\vec{C}^i$ which will contain the latest i positions from $\vec{P}$.

2. Then, we create a matrix of $m \times i$ elements, and fill it with the distances between each point from $\vec{G}$ and $\vec{C}^i$.

3. Starting at the bottom-left element of the matrix (which is equivalent to the distance between the first element of $\vec{G}$ and the first of $\vec{C}^i$), we select the path to the top-right element with the smallest cost.

4. If the cost of the path is not greater than a established threshold th, it would mean that $\vec{C}^i$ matches $\vec{G}$, and thus, we have detected a hand gesture.

Conceptually, the dynamic time warping algorithm consists in matching every point of $\vec{C}^i$ with the most similar one of $\vec{G}$, as Fig. 3.3(left) illustrates. In this same figure (right), we can see an example of a warping matrix which matched two samples: there is a low cost path between first and last elements (shaded in grey).

This solution has proved to be a reliable one in many fields which require a flexible matching of two temporal sequences. However, in our case the DTW algorithm did not perform very well: it had a high recognition ratio of the gestures, but also many false positives from random movements of the hands.

Finite state machines

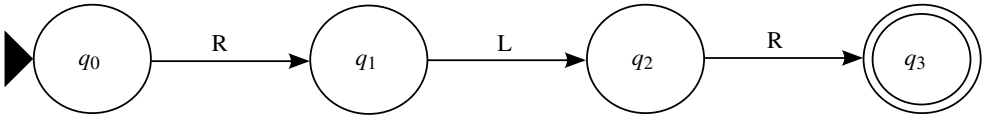


Figure 3.4: Finite state machine for recognising the wave off gesture. See text for a formal definition on the transitions and the states.

Finite state machines can be used to recognise hand movements [63]. For each hand movement that we want to recognise, we need to define a set of states and rules to transit between states. Fig. 3.4 illustrates the FSM that we have designed for the wave gesture. Formally, it is represented by a 5-tuple (Q,Σ,δ,q_0,q_3) where:

- The set of states is defined by $Q = \{q_0, q_1, q_2, q_3\}$.
- $\Sigma = \{R, L, N\}$ is the alphabet of our machine.
- δ is the transition function.
- q_0 is the initial state, and q_3 the acceptance state.

Given that what we want to analyse with this FSM are the incoming points from the hand tracker, we need to translate them into symbols of the alphabet of our machine. For this reason, we execute the following procedure:

1. We create sets of points from each trajectory provided by the hand tracker. A set of points S will be composed of consecutive points which describe a movement of the hand in the horizontal axis (positive or negative), that is, we add points to our set until there is a change in the horizontal direction. In order to filter out noise in the hand tracking, the change must be greater than 5cm.
2. If S represents a straight line of at least d cm long and covered in less than t time:
   - We assign the symbol R to those segments with positive horizontal movement.
   - We assign the symbol L to those segments with negative horizontal movement.
3. If the conditions on the previous rule are not met, the segment will be represented with symbol N.

In our case, we use $d = 7cm$ and $t = 500ms$. Moreover, we evaluate the criteria of *fitting a straight line*, by fitting $\vec{P}$ to a line with the least squares method, and computing the average error to this line.

In this way, hand movements which contain the pattern *RLR* are recognised by our FSM (Fig. 3.4) as wave gestures, while those random movements of the hand will contain N symbols, and thus they will be discarded.

Similarly to the wave gesture, we have also designed an FSM for each of the *swipe-left*, *swipe-right* and *push* gestures. Essentially they are simplifications of the one that we have just described, i.e. the the swipe-right gesture can be described with states q_0 and q_1 and minor modification on the rules that create symbols (increasing d to 40cm).

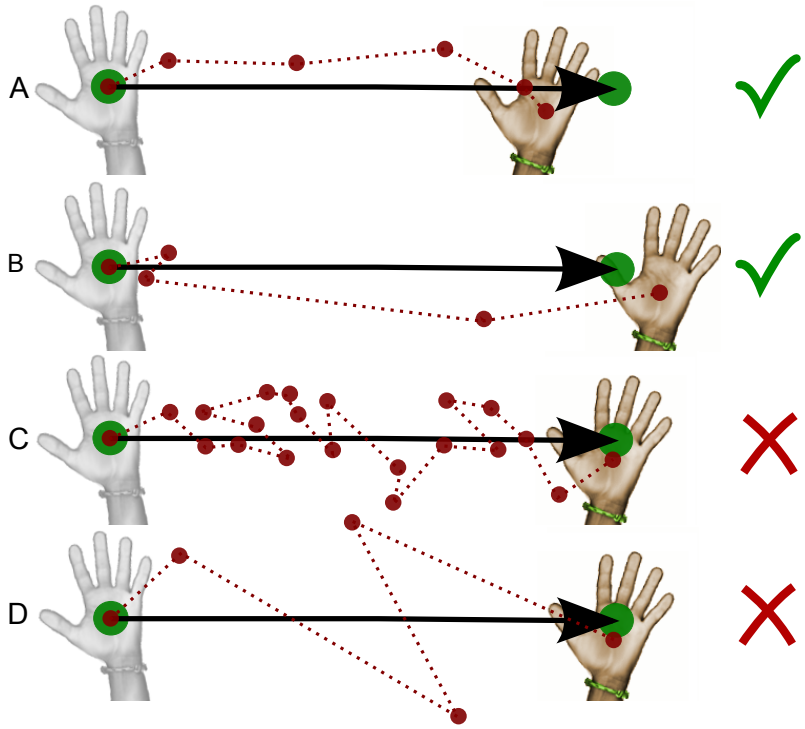


Figure 3.5: Different examples of gesture recognition and rejection for the swipe-left pattern (simplified in the 2D space). The gesture start and end points are drawn with green points. The gesture path is represented with the dotted line and the red points. The black arrow simulates the straight line that fits the points.

This approach also introduces speed tolerance and certain flexibility in the way that a human should perform the gestures, as we illustrate with the examples of Fig. 3.5. This figure shows four trajectories and the outcome of each one when it matches a swipe-left gesture. Examples A and B are accepted as swipe-left gestures showing certain flexibility in the recognition: different number of points and even small changes of direction (noise). Example C is rejected because, even though the hand trajectory could fit an straight line, it is performed very slowly, and thus could be a random movement of the hand. Example D is rejected because the trajectory of the hand does not fit a straight line.

Conclusions

We have decided to use the finite state machines over the DTW because they offer a better interpretability of the recognition process, allowing us to adjust it to the gestures that the users of our robot usually perform. Moreover, we were able to sort out the false positives that we obtained with the DTW.

Additionally, the hand detection, tracking and movement recognition processes that we have just described are able to run in less than 30ms. This aspect is very important for a robot like the one we have developed because it has to run many processes in parallel, sharing computing power between all of them. A fast processing of the hands makes it very unlikely that we lose the hands owing to a rapid movement. Moreover, we use only depth images from the robot's camera. This means that we are able to detect and track the hands in any environment, regardless of the background colour since there is no interference between it and the hand tracking process, which is an extended drawback in previous hand detection and tracking solutions.

3.3.2 Augmented reality graphical user interface with voice feedback

In order to help the user ease and guide his communication with the robot, we have designed an augmented reality graphical user interface (AR-GUI) with virtual buttons which can be activated using hand movements. The AR-GUI will provide on-screen information and voice feedback to keep the user informed about what the robot is sensing and what he expects from him at every time. The AR-GUI is the users' window to what the robot sees, thinks, and expects from him.

The communication scheme of our robot consists in the necessary steps that a user should perform to interact with our robot. In order to achieve an usable and efficient design, we have

carried it out in several stages. Each of these stages consisted of a candidate AR-GUI, a set of candidate commands and a set of rules. In each stage of the design, we have gathered feedback from many users by means of a quality-in-use test. This approach has enabled us to evolve the original design into a more user-friendly communication tool. In this section we will describe the final version obtained from this process, while the description of how the design evolved is explained in section 3.3.3.

Currently, the augmented elements of the AR-GUI are:

- The human body is shaded with a random colour when he is being detected by the robot. Moreover, in case that a human is the robot's target (i.e. when following him), his body is shaded in green).
- When a human is performing the initialisation pose to let the robot detect his hands (Fig. 3.2), two red boxes will pop up, to indicate where should the user put his hands.
- Blackout of the areas not visible for the robot: i.e. if a human is outside the working range for the robot's human detector, he will not appear in his screen.
- Two virtual balls, drawn over each hand while they are detected and tracked by our robot.
- Virtual buttons to command actions to the robot: i.e. the instructor of our robot is able to start recording a new route by holding his hand over a virtual button for a couple of seconds.

When thinking about which gesture should command which action in our tour-guide robot, we have tried to make associations that humans can regard as natural. For example, in most cultures it is widely accepted that a *wave-off* gesture means either to welcome or to say good-bye to someone, therefore we perform that gesture when we want the robot to stop interacting with us. Another example would be that whenever we want someone to follow us, we tend to move one or both hands towards us. In our robot the action of joining both hands together is associated precisely with that action: the robot will start following us. However, there are many actions which lack off such a widely accepted association. That is the case of 'record a route' or 'stop recording my voice message'. In order to overcome this problem we have added a virtual button for each of those actions on the robot screen.

Through this AR-GUI the user can notice if a gesture has been recognised or if he is being detected and recognised by the robot, for example. Thus the user quickly learns to move

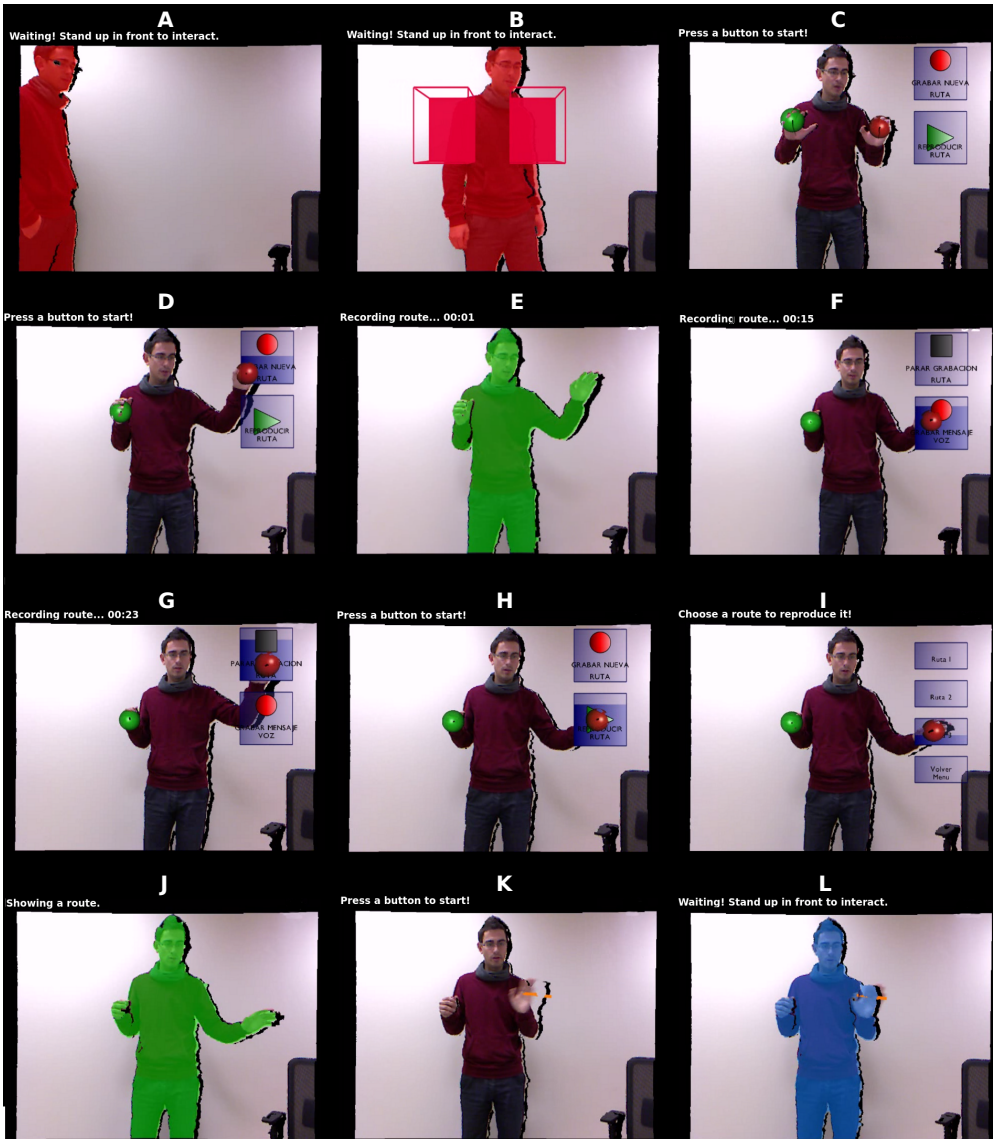


Figure 3.6: Augmented reality graphical user interface (AR-GUI) displayed on the robot's screen. We can see how a user enters the robot's field of view, initialises his hand positions and starts giving commands to the robot.

in a certain way so that the robot does not lose him. In Fig. 3.6 we can see a sequence of screenshots from the AR-GUI taken when performing a typical interaction with our guide robot. First, when a human walks into the robot's field of view (Fig. 3.6.A), his body is coloured in a random colour distinct from green. Then, when that user is standing in front of the robot a voice message asks him to *please, put your hands up towards the robot*, and immediately two red boxes are drawn at the positions where the user should put his hands (Fig. 3.6.B). This step helps a robot's user to correctly perform the initialisation pose.

When the robot has detected the user's hands, the hand positions are marked with two virtual balls. At the same time, two buttons (one for recording and another for reproducing routes), appear on the screen (Fig. 3.6.C), and a voice message requests the user to select an action. Then, the user of our example activates the *record a new route* button by holding his hand over it for a couple of seconds, as shown in Fig. 3.6.D. At that point, the AR-GUI shaded his body in green (Fig. 3.6.E) and the robot starts following the user around the environment. Again, the user receives information via voice about the action: the robot is now *recording a route*. Now, at any point, the user might decide that he wishes to record a voice message (when a visitor selects this route for reproducing, this voice message will be played back at the same point where it was recorded). In order to do this, the user has to stop in front of the robot and initialise his hand positions (as shown before in Fig. 3.6.B and Fig. 3.6.C). Now, since the user is already recording a route, the buttons which appear on the screen are to stop the route recording process and to record a voice message. Thus, the user activates the button to record a voice message, as shown in Fig. 3.6.F, records his message speaking loudly towards the robot, and then stops the recording of the message by pressing a button again. The message will be reproduced immediately afterwards to offer the user a chance to record it again. The voice message will be stored with the route information, and can thus be reproduced at the same point of the route where it was recorded. Once the user has taught the new route to the robot, he activates the *stop the route recording* button (Fig. 3.6.G), and the information on the route is stored in the robot, so that the robot can reproduce it afterwards.

Once the robot has learnt several routes, it can offer its guidance services to anyone who stops in front of it, and asks for a route to be reproduced by activating the button as shown in Fig. 3.6.H. This button leads to a sub-menu with as many buttons as routes are available (Fig. 3.6.I), and a voice message asking to *choose one route* is played from the robot while the routes are being shown. When the user has chosen a route, the robot will finish the hand interaction (Fig. 3.6.J) and start reproducing this route. Finally, when a user decides that he

no longer wishes to interact with the robot, he can wave off (after proper initialisation of the hands position). A voice message saying *goodbye* is reproduced and his body will be appear shaded in a colour other than green. This process is shown in Fig. 3.6.K and Fig. 3.6.L.

Moreover, throughout the entire period of interaction a text message appears in the top of the screen with information about the robot's state. Also, it is important to note that our guide robot allows humans to walk freely. In the case that the user is not looking at the robot's screen, and is walking too fast, he is informed by a voice message to *please, walk slower*. Also, if the user walks outside the working range, he is informed with a voice message saying, *I lost you*, and then the user is responsible for coming back to be recognised by the system.

3.3.3 Usability tests

In this section we want to measure the quality in use achieved with the interaction framework that we have designed. Hence, adhering to the ISO IEC-9126-4 Standard, we have conducted a user study consisting of a survey and the measurement of several parameters during each user test. The experiments were carried out in several test-rounds and with a heterogeneous set of users.

The ISO/IEC-9126 is an international standard for the evaluation of software quality. More specifically, the fourth part of the standard (ISO IEC-9126-4) stipulates four quality in use metrics, and the recommendations on how the evaluation should be carried out. How metrics are collected depends on the product that we want to evaluate. Therefore, following the suggestions of the ISO standard, and given that our software product consists of a robot behaviour, the following is a description of how we have considered each one of the four metrics:

– Effectiveness. Monitoring a user during tests can provide us several effectiveness measurements, one of the most widely used is error frequency: how often does the robot make a mistake?.

– Productivity. Efficiency and task time should be measured while performing the tests with the guide robot. In our case the participants were given a 10 minute time limit to perform the set of actions. Despite this point was not being our main concern, all the participants in our test were able complete the test on time.

– Safety. It is quite important to develop a safe behaviour for our robot, therefore reports on safety issues or software damage are included.

– Satisfaction. Interviews or questionnaires are the recommended way of obtaining psychometric values of the user's perception and have also been used in other works [66, 41, 76] as the main source of information.

During the test that we describe in the following sections, the robot was in operation for about 15 hours in four different events, mainly outside our department and in real world scenarios, such as in a engineering school, secondary schools or even a museum. Several test and correction stages took place in between the events, as we detail below.

First test in an academic environment

The first round of tests involved of 12 users with different backgrounds: there were six undergraduate students (three from biology, one from computer science and one from medicine), a graduate in political and social sciences, and five Ph.D students. There were three females and nine males, and their average age was 26 years (20 to 34). This round of tests was carried out in our department and lasted for about 2 hours.

Before starting the tests we gave the following information to the users: a) our robot is controlled by hand movements and virtual buttons, and b) it is necessary to perform a initialisation pose before it can recognise any gesture. After that, they were asked to perform the following sequence of actions:

1. First, test the following behaviour of the robot. Order it to start following you, and walk around.
2. Stop and command the robot to record a route by pressing the button in the menu. The robot will not start following you yet.
3. Order the robot to follow you while recording the route by joining your hands. Now, move around as you wish.
4. Stop and order the robot to record a voice message by pressing the voice record button that appeared after pressing the route record button. When you are done, stop the voice recording by pressing the appropriate button.
5. Stop the route recording by pressing the corresponding button.
6. Finally, finish the interaction with the robot by waving off.

Gesture	Avg. trials	Std. Dev.
Init hands #1	1.00	0.00
Join hands #1	1.00	0.00
Init hands #2	1.00	0.00
Swipe L/R #1	**1.75**	0.62
Press route record	**1.58**	0.79
Join hands #2	1.08	0.28
Init hands #3	1.00	0.00
Swipe L/R #2	**2.33**	0.78
Press voice record	1.25	0.62
Press stop voice record	**1.42**	0.51
Press stop route record	1.25	0.45
Wave	1.17	0.58

Table 3.1: Average trials measured for each gesture in the first test that was carried out by twelve individuals. Bolded numbers indicate gestures with high error ratios (> 1.25).

At the time of this test, in order to carry out this sequence of actions it was necessary to perform the twelve gestures shown in the left column of table 3.1. Moreover, the right columns of this table show the average number of trials until a gesture was successfully performed by those who tested the robot. In order to capture this data, we have recorded the whole tests in video, and then manually counted their trials. We can observe that the gesture that gave rise to most problems for recognition is the left/right swipe gesture with an average of 2.33 trials before the robot could recognise the hand movement. The second gesture with most problems is the hand movement used to press a button with an average of 1.42 trials. These results can already gave us clues about the drawbacks of our first design. However, to go into more detail about those weakness, we handed out a small survey to the participants of our tests.

In the survey, we asked the participants about the agility in handing down a command with each gesture, and also about three aspects of the experience in general:

– How did they feel about the interaction with gestures.

– How did they feel about the interaction with virtual buttons.

– How did they feel about the performance of the person following behaviour.

User ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	Avg
Init pose	3	5	4	4	5	4	4	3	4	4	5	4	4.00
Join hands	3	5	4	4	4	4	4	5	5	5	4	4	4.27
Swipe L/R	5	3	3	3	3	4	5	3	3	3	4	3	3.27
Push a button	1	1	2	-	2	1	-	2	4	4	3	2	**2.22**
Touch a button	5	4	3	3	5	4	4	5	5	5	5	4	4.36
Goodbye(wave)	4	5	4	5	5	4	5	5	5	5	2	4	4.45
Hand gestures	4	4	4	4	3	4	4	4	4	5	4	5	3.82
Button interaction	1	2	3	4	4	4	2	2	2	5	2	3	**2.91**
Person following	3	5	4	3	5	3	3	4	5	4	5	4	3.73

Table 3.2: Average and individual ratings obtained from the first survey that was handed out to twelve individuals in our department. In the case of the push gesture that is used to press a button two individuals did not try it. Bolded numbers highlight the results below the the average rating (<3.0).

The opinions for each item were given through the five star scale: where 1 star means that they disliked the performance or usability of the item, while an 5 stars means that they had no problems and felt that the quality in use of that item was excellent.

Table 3.2 shows the ratings obtained from each individual, as well as, the average value for each item. From these ratings, we can conclude that the users could quit easily wave off in order to say *goodbye* to the robot, and also that they could *touch-to-press* the virtual buttons in our AR-GUI. The users also valued the initialisation pose: comments about it said that it was not difficult to perform, and even that it was quite natural when calling someone's attention. Similarly to the results shown in table 3.1, the worst mark was obtained by the left/right swipe gesture to show/hide the menu, and also by the hand movement to press a button (referred as *push a button* in table 3.2). It is also worth mentioning that the users would choose hand movement gestures instead of an interaction based in virtual buttons. We believe that this is a consequence of the poor quality of the recognition of the push gesture for pressing buttons. Finally, we want to remark that the person following behaviour was also evaluated positively as an important contribution of our tour-guide robot.

Complementing this view, it was very useful to listen to the users' comments during this test, as well as to read those collected in the comments section of the survey. The following list summarises the conclusions that arise from them:

– One of the key abilities of our robot is the person following ability. Despite that it got a high rating (3.73), we expected a higher one. We found the reason for this rating in the

comments: “when following, the speed was not fast enough to keep with their walking speed”. However, when asked about positive aspects of it, the users valued positively the ability to be recognised again in cases where the robot had lost them.

– When performing the left/right swipe gesture the users tended to connect both hands just before starting the hand movement. This caused the robot to recognise the gesture (’follow me’) and ignore the next one. One user even proposed a solution for this: the join hands gesture should only be activated when the hands are connected for a few seconds.

– When pressing the virtual buttons on the screen most of the users decided to just hold their hand over the button for a second to press it. They claimed this was a fast and easy option for them, as they knew that they were doing it well because the button started to fill its colour. Nevertheless, some users decided to try to use the push gesture, and while half of them succeeded, the other half could not make it work, which explains the bad rating obtained by the ’push a button’ in contrast to the rating obtained by ’touch a button’ (table 3.2). The users who were unsuccessful on using the push gesture switched to the other method of pressing a button. This is why the number of trials for pressing a button for the first time was 1.58 and then it fell to 1.25, which means that few users kept trying to push the virtual buttons instead of just *touching* them (holding the hand over them).

– At certain points of the interaction, the users did not know what they had to do. Some users suggested that it would be a good idea to introduce a help button or to increase the number of voice messages in certain situations and to use a voice clearer than the one the robot currently had. On the other hand, three users felt that the voice message that is reproduced by the robot whenever the user is in front of it (’raise your hands towards me to start interacting’) should not be played that much: they felt it was a bit annoying.

– If the robot speed is high, and the user stops, the robot is unable to stop at a suitable distance for the interaction. This meant that some gestures were partially performed outside the camera’s field of view, which lowered their recognition ratio. The users suggested that the robot should step back in such cases.

– Other suggestions include: a) the removal of the initialisation pose, b) the increase of the number of animations and contrast of the virtual interface, c) the search for a more

natural 'follow me' gesture than joining hands, and d) the improvement of the robot's appearance.

Finally, after this first round of tests we corrected certain aspects of our system in line with the feedback collected here. The changes are listed below:

- The speed was increased sightly up to the hardware limit.
- The join hands gesture used to command the robot to follow the user was slightly changed following a suggestion from a user. This gesture now requires to connect both hands for at least three seconds before the robot recognises it as a *follow me* command.
- The voice message informing a user to *raise his hand up towards the robot* was restricted to fewer situations. Now it is only reproduced when the robot has remained at standstill for three seconds.
- The voice of the messages was changed from text to speech software to pre-recorded messages by a woman.
- The distance from the robot to the person has been increased sightly to allow the robot to stop at a proper distance when the person is moving fast and suddenly stops.
- The recognition of some gestures was adjusted to improve the recognition rate. Gestures like wave might present high variations in form, size and speed depending on the person who is interacting with the robot.

Second Test at the Engineering School

The second round of tests was designed to test whether the first changes that we made to the interface were along the right lines. The tests and surveys were exactly the same as in the first round, in order to be able to compare the results. The main difference was that here, the tests took place in an engineering school in an open event where volunteers could come and test our robots for around 4 hours. Thus, the participants were chosen from among the public that came to the exhibition, although due to the high affluence of public at certain times, the majority of the people who tested it could not take our survey. At times with less audience, we managed to collect the opinions of seven people.

In table 3.3, we can see that the gesture that was most difficult to recognise was, once again, the left/right swipe gesture with an average of 2.29 trials (for the first time that it had to

Gesture	Avg. errors	Std. Dev.
Init hands #1	1.14	0.38
Join hands #1	1.33	0.52
Init hands #2	1.00	0.00
Swipe L/R #1	**2.29**	0.49
Press route record	**1.43**	0.79
Join hands #2	1.00	0.00
Init hands #3	1.00	0.00
Swipe L/R #2	**1.57**	0.53
Press voice record	**1.43**	0.53
Press stop voice record	**1.43**	0.53
Press stop route record	1.29	0.49
Wave	**1.43**	0.53

Table 3.3: Average trials measured for each gesture in the second test that was carried out during an exhibition in an engineering school.

be performed) and 1.57 trials (for the second time that it had to be performed). This reduction in trials between the first and the second time is best explained with one comment collected from a user: *you need to learn how the gestures should be performed, once you know that, your gesture is easily recognised*. The second most problematic gesture was the one used to press a button, with an average of 1.43 trials. Once again, the users who tried to push a button made more errors than those who have chosen to just hold their hand over the virtual button.

User ID	#1	#2	#3	#4	#5	#6	#7	Avg
Init pose	4	5	5	5	5	3	4	4.43
Join hands	5	5	5	5	-	5	5	5.00
Swipe L/R	5	4	4	5	3	4	4	4.14
Push a button	2	2	3	3	3	2	2	**2.43**
Touch a button	5	5	5	5	5	5	5	5.00
Goodbye(wave)	5	5	4	4	4	4	4	4.29
Hand gestures	5	4	5	4	3	4	5	4.29
Button interaction	1	3	5	2	4	4	4	3.29
Person following	5	4	4	5	4	3	5	4.57

Table 3.4: Average and individual ratings obtained from the survey that was handed out to 7 persons at the Engineering School. In the case of the join hands gesture that is used to command the robot to follow the user, number #5 did not try it. In bold we highlight the results below the average (<3.0).

In the same way than in the first test, table 3.4 shows the psychometric values obtained from each individual, and the average value for each item. This time, the users evaluated two tasks with the highest mark: joining their hands to command the robot to follow them, and touching a virtual button in order to push it. It was important for us to confirm that the changes that were introduced in the first round helped to increase the quality perception of the users. The initialisation pose and goodbye gesture were still amongst the highly valued items. It is also worth of noting that the rating of the person following ability increased by almost one point: we had increased the maximum speed of the robot and many people considered it quite important when evaluating the person following behaviour. Also, it is likely that the improvements in the recognition of the *follow me* command are responsible for the increment in the rating of the person following behaviour. Nonetheless, the button interaction is still attached to a medium rating, which is a consequence of the combination of the bad behaviour of the push gesture with the good performance of touching a virtual button.

In the same way than in the previous round of tests, we have decided to improve several aspects of our robot, using user's comments. The changes that we did to our robot are listed below:

- The working range of the laser leg detection algorithm was increased to detect legs at greater distances from the robot, correcting some small bugs in the process.
- The tracking has been modified to include an improved prediction of human positions based on the robot's odometry.
- The route recording process was modified to simplify the process:
  - Now, when the *record route* button is pressed, the robot immediately starts to follow the user while recording a route. Prior to this change the user had to perform the *follow me* command after pressing the button to record a route.
  - The left/right swipe gesture to show and hide the menu has been removed. Now, the menu is automatically shown only when the user is actually going to use it. In other cases the menu is hidden, for example while the robot is following the user.

Tests at Schools and showcases at the Domus Museum

After these two test-correction rounds, we have conducted out a third set of tests. First, we have showcased our robot at a local secondary school, then we went to a local elementary

school and finally, we did an initial test and a showcase at the Domus Museum in the city of A Coruña (Spain). This scientific museum allowed us to test our robots in a challenging environment.

Our first event was at a secondary school. In this location, around 40 different people, with ages ranging from 15 to 18 were able to test and interact with the guide robot for almost three hours. First, we gave a brief explanation and demonstration to four groups of around 25 people, and then about 10 volunteers from each group tested the robot themselves. They commanded the robot to follow them and tested its recognition abilities. The next event (elementary school) was quite similar but this time with younger people. There, around 50 boys and girls, whose ages ranged from 9 to 10, were able to test the robot themselves for around 4 hours.

In both the elementary and the secondary schools, children showed a high interest in watching and testing how the robot could follow them. Unfortunately, due to the amount of people attending these events, and the nature of the same, we were not able to hand out any surveys to properly monitor the users. Nevertheless, the comments from these users were still valuable for us. Amongst the many comments received we want to highlight the following: '*the robot should move faster to follow a human at normal walking*', and '*the robot should be in a human-like shell, the shelf-like appearance of the robot is not appropriate*'. Both comments suggest us that now, those who could test it are more worried about its appearance than about his mistakes, or bad performance. These tests also allowed us to test our robot in really crowded spaces, which in some cases resulted on a slower behaviour, although in all cases the robot succeeded on its task.

The last place where we have test the human-robot interaction of our robot and its person following abilities was the Domus museum. Its floor is made of raw slate and thus quite uneven. There were several light sources from panels with different colours, alongside two large, naturally lighted areas. These special characteristics were challenging for our robot and its mostly vision-based recognition abilities.

In this environment, we first tested the guide robot ourselves for around three hours, walking around and making sure that the robot was capable of working in any part of the museum, regardless of the special light conditions. In that time, we identified several flaws in the robot's software as well as some fine tunning of the recognition system, necessary to work in these conditions. We returned to our laboratory and while correcting the minor problems found during this initial test, we organised an open event in collaboration with the museum for whole

day to showcase our robots. In that event, the robot was in operation for about 6 hours. The event was restricted to forty people who required to sign up for the event, and it was full in just some hours after the museum announced it. Everyone could test the robot: they could command the robot to follow them inside an small circuit. Children specially liked the robot, they fought for the robot's attention and even claimed that he seemed to be alive. In the end, the robot was successfully showcased, the general opinion was very positive, and the number of robot's mistakes, i.e., confusing the target with a distractor, was quite low.

As a summary, during the tests that we have carried out, people maintained a proactive attitude towards the robot. They found highly positive that the robot would keep them informed about its actions and problems, which made the interaction easier. That suggests us that we are in the right direction to obtain a tour-guide robot usable by anyone without previous expertise in robotics.

3.4 Conclusions

In this chapter we have described the human-robot communication scheme of our tour-guide robot. The main strengths of our robot are: its easy and natural interaction through gestures and virtual buttons, which are shown in an augmented reality graphical user interface, and its friendly visual and voice feedback mechanism.

The purpose of the AR-GUI is to allow humans to cooperate and interact easily with the robot. In order to test the usability of our robot we have adhered to the ISO/IEC-9126-4 standard to measure quality in use, by monitoring users and handing a brief survey to collect their opinions. These users were randomly picked in a event at the local engineering school and in our department. We have also tested it at a elementary school, at a high school, and at a scientific museum. In total more than 150 people were able to interact with our guide robot in the tests that we described in this chapter. From these tests we were able to pinpoint the pros and cons of our robot interface. In light of these results, we accomplished several changes to overcome the problems encountered in the communication of our tour-guide robot. In the end, the most frequently mentioned weakness is the robot's speed, while the positive aspects of our robot were: the human recognition ability, and the fast and easy interaction achieved with our AR-GUI interface.

CHAPTER 4

ROUTE RECORDING

Route recording is the process of introducing route information in a robot: the trajectory, a description of a point of interest, etc. Traditionally, this information is introduced in tour-guide robots by experts in robotics. This fact significantly increases the deployment time and cost of such robots, which represents an obstacle towards the goal of bringing robots to everyone's home or office. In the tour-guide robot that we are developing, we allow the robot to learn new routes while following an instructor. It is important to mention that the instructor could be anyone with few knowledge about robots, i.e. an staff of the event where the robot operates.

We have already described two important parts of our proposal: the following of a human and the recognition of his commands. However, the route recording behaviour is not yet complete. In order to complete it, the robot should be able to estimate its position in the environment, as well as, to store that position (along with other information) to build the route. The task of estimating the robot's position is still an open problem in unstructured environments, which might be full of people moving around the robot. In this regard, apart from describing our route learning architecture, an important part of the content of this chapter consists in the description of our novel localization algorithm that provides the route recording with a global estimate of the robot's position. The localization algorithm was designed and implemented in colaboration with Adrian Canedo Rodriguez.

4.1 Challenges in route recording

The problem of recording the route path requires the robot to be able locate itself in the environment. In this regard, one of the main challenges of our tour-guide robot is to accurately locate itself in the indoor environment where it is operating.

Indoor environments usually contain several places which are quite similar to others in the same building. A good example of this situation are offices in a office building, or several floors of the same building. This situation is usually referred as symmetries and represents a major challenge in indoor environments. Another important challenge in robot localization is related to the non-stationarity character of environments. That is, the robot should not use as reference for its localization objects that can easily change its position with time. A clear example of this problem is furniture: the position of chairs and tables is often changed. Finally, a third important challenge of current robot localization solutions is that the reference that they use might be temporary occluded. In this regard, humans are a challenge themselves: service robots operate in locations where many humans can partially occlude their view of the surroundings, or even change the properties of the environment (moving furniture or attenuating signals that help the robot to compute its position).

In our tour-guide robot, a robust robot localization is of key importance: if the robot does not record the correct path while it follows the instructor, he will need to repeat the process.

4.2 State of the art

In the late nineties, the first well-known tour-guide robots, *Rhino* [2, 4] and *Minerva* [5], had no online route learning abilities. In fact, route information was introduced in the robot by an expert. This has been a common element in the tour-guide robots that were developed afterwards. Examples of those include: *RoboX*[77], which was a tour-guide robot designed for long time operation in a public exhibition where the routes were introduced by experts before the exposition, and Robotinho [78], which was one of the first humanoid tour-guide robots. Its routes were also manually introduced in a pre-operational stage of the robot.

In recent years, some robots have started to introduce some novelties in this topic. Thus, although Urbano [12] is a tour-guide robot which requires a manual creation of a navigation graph, and a database of objects and locations. However, this information allows the robot to dynamically generate routes that will be shown to a visitor depending on user's category. Another example, a recent tour-guide robot [79] allows the user to select which exhibit he

wants to visit using speech recognition, and can also identify which part of the exhibit needs further explanation by recognising pointing gestures.

Despite of these improvements in route management, these tour-guide robots still require a heavy pre-operational stage in which an expert introduces locations and other data about the exhibits available. The only case that we have found that might introduce certain route learning abilities is a recent tour-guide robot [67] that states that new routes can be created on the fly. Unfortunately no details about the process are given.

Contrary to the route recording problem, mobile robot localization in indoor environments has been deeply studied in the past years. The researchers behind the first tour-guide robots Rhino and Minversa laid the foundations of one of the most popular approaches nowadays: the MonteCarlo Localization (MCL) algorithm [80]. It consisted in estimating the robot's pose by integrating laser information in a particle filter. Later on, new sensors were also integrated in this algorithm which proved to be able to cope with sensors dealing noisy measurements (i.e. sonars [81], cameras [82]). In the next years, the popularity of particle filters for robot localization kept growing with improved versions of the original MCL algorithm.

4.3 Multi-sensor algorithm for mobile robot localization

Generally, localization algorithms use only one sensor, which may be a problem, specially when: a) the sensor data is highly noisy, b) the sensor fails to provide data, c) different areas look alike to the sensor, etc. These problems get worse in crowded environments, because there are people moving around which produce occlusions or changes in the environment like the moving of furniture. In order to address these issues, we have proposed [83, 84] a localization algorithm that combines the evidence supplied by several sensors.

The solution that we propose is based on the Augmented Montecarlo Localization Algorithm (AMCL), and unlike other solutions [85] (e.g. Kalman Filters), it can properly handle: a) sensors with non-Gaussian noise, b) non-synchronised sensors, c) sensors with different data rates, or d) sensors that can stop providing data (e.g. sensor failures).

Therefore, the goal of our solution is to estimate, at any time t, the robot's pose $\vec{s_t}$ using: perceptual information $\vec{Z_t}$ (or the set of sensor measurements), and control data u_t (the robot movement as provided by odometry encoders). The robot's pose is defined by $\vec{s_t} = (x_t, y_t, \theta_t)$, where x_t and y_t are the position coordinates, and θ_t is the orientation one.

Essentially, in order to accomplish our goal, we have to previously compute the likelihood $bel(\vec{s}_t)$ of every possible pose of our robot, using u_t and $\vec{Z}_t$ (we call this process *pose probability estimation*, section 4.3.1). Then, we perform a *pose estimation* process that from these likelihoods computes the robot's pose $\vec{s}_t$ (section 4.3.2).

4.3.1 Pose probability estimation

Following a Bayesian filtering approach [86], the likelihood assigned to each robot pose $bel(\vec{s}_t)$ will be the posterior probability over the robot state space conditioned on the control data u_t and the sensor measurements $\vec{Z}_t$:

$$bel(\vec{s}_t) = p(\vec{s}_t|\vec{Z}_t, u_t, \vec{Z_{t-1}}, u_t, ..., \vec{Z}_0, u_0) \tag{4.1}$$

Assuming that the current state $\vec{s}_t$ suffices to explain all the previous states, measurements and control data (Markov assumption), we can estimate $bel(\vec{s}_t)$ recursively [86]:

$$bel(\vec{s}_t) \propto \left[\int p(\vec{s}_t|\vec{s_{t-1}}, u_t) bel(\vec{s_{t-1}}) d\vec{s_{t-1}} \right] p(Z_t|\vec{s}_t) \tag{4.2}$$

In this equation, the term $\int p(\vec{s}_t|\vec{s_{t-1}}, u_t) bel(\vec{s_{t-1}}) d\vec{s_{t-1}}$ is in charge of inferring the new $bel(\vec{s}_t)$ from $bel(\vec{s_{t-1}})$ and u_t. On the other hand, the term $p(\vec{Z}_t|\vec{s}_t)$ corresponds to the update process in charge of sensor fusion, where $\vec{Z}_t = \{z_1, z_2, ..., z_{n_t}\}$ is the set of all sensor measurements at time t (n_t is the number of sensors modalities available at time t). Therefore, $p(\vec{Z}_t|\vec{s}_t) = p(z_1, z_2, ..., z_{n_t}|\vec{s}_t)$ represents the probability that, at time t, the system receives the sensor measurements $\{z_1, z_2, ..., z_{n_t}\}$ conditioned on the state $\vec{s}_t$. This joint probability function may be very hard to estimate in practice, specially if our sensors provide information at different data rates. For this reason, we assume that the sensor measurements are conditionally independent given the state of the robot, therefore:

$$bel(\vec{s}_t) \propto \left[\int p(\vec{s}_t|\vec{s_{t-1}}, u_t) bel(\vec{s_{t-1}}) d\vec{s_{t-1}} \right] \prod_{k=1}^{n_t} p(z_t^k|\vec{s}_t) \tag{4.3}$$

In order to be able to apply Equation 4.3, we must know: 1) the initial belief distribution $bel(s_0)$ (it can be chosen randomly), 2) the *motion model* of the robot $p(\vec{s}_t|\vec{s_{t-1}}, u_t)$, and 3) the *measurement model* $p(z_t^k|\vec{s}_t)$ of each sensor k. The motion model represents the probability of transition from the state $\vec{s_{t-1}}$ to the state $\vec{s}_t$, provided u_t. This model depends on the odometry of the robot, but it is common to assume that it follows a multivariate normal distribution [86]:

$$p(\vec{s_t}|\vec{s_{t-1}}, u_t) \sim \mathcal{N}(f_{mov}(\vec{s_{t-1}}, u_t), \Sigma_s) \tag{4.4}$$

where f_{mov} is a function that models the movement of the robot, and Σ_s represents the sensor noise covariance matrix of the model. On the other hand, the measurement model $p(z_t^k|\vec{s_t})$ depends on the nature of each specific sensor. This will be described in detail in section 4.4.

Estimation with a particle filter

Equation 4.3 can be approximated very efficiently using a particle filter [86]. In this regard, $bel(\vec{s_t})$ can be approximated with a set of M random weighted samples or particles:

$$bel(\vec{s_t}) \approx \vec{P_t} = \{\vec{p_t^1}, \vec{p_t^2}, ..., \vec{p_t^M}\} = \{\{s_t^1, \omega_t^1\}, \{s_t^2, \omega_t^2\}, ..., \{s_t^M, \omega_t^M\}\} \tag{4.5}$$

where each particle $\vec{p_t^i}$ represents a robot state hypothesis s_t^i at time t and its likelihood ω_t^i. We estimate Eq. 4.3 using the Augmented Montecarlo Localization algorithm (AMCL) with low variance resampling [86]. Essentially, this algorithm proceeds as follows:

Initially, all particles are distributed randomly over the state space, and assigned equal likelihoods ($\frac{1}{M}$). Then, the algorithm performs the following operations:

1. Propagation. Each particle is assigned a new state according to the motion model of our robot (described by Eq. 4.4):

$$\vec{s_t^i} \sim p(\vec{s_t^i}|\vec{s_{t-1}^i}, u_t) \quad \forall\, i \in \{1, 2, ..., M\} \tag{4.6}$$

2. Importance factor. The weight of each particle is updated given the available sensor measurements:

$$\omega_t^i \propto \prod_{k=1}^{n_t} p(z_t^k|\vec{s_t^i}) \quad \forall\, i \in \{1, 2, ..., M\} \tag{4.7}$$

3. Resampling. The algorithm creates a new particle set $\vec{P_t}$ from the previous one $\vec{P_{t-1}}$. First, it draws M_{NR} particles from $\vec{P_{t-1}}$ with a probability proportional to their weight ω_t^i. We use the Low Variance Resampling technique [86] to accomplish this task. Second, it generates a variable number of random particles (M_R), to increase the robustness against problems such as the *robot kidnapping problem* [86]. Finally, the union of the M_{NR} non-random and the M_R random particles ($M_{NR} + M_R = M$) will form the new particle set $\vec{P_t}$.

4.3.2 Pose estimation

After computing $\vec{P}_t$, we need to estimate the pose of our robot. There are many methods to estimate $\vec{s}_t$ from $\vec{P}_t$: a weighted mean over all the particles is one of the most popular. However, in some cases methods based on simple analytics like that one might deal strange results, i.e. if we have two bulks of particles with similar weights in different positions it will estimate that the correct position is between these two sets. In order to avoid these situations, we have decided to perform an agglomerative clustering operation following a hypothesis selection.

Therefore, first, we assign each particle in $\vec{P}_t$ to the cluster with the closest centroid, provided that it is closer than a threshold distance max_{CD}. Otherwise, the particle will create its own cluster. In this way, we will obtain one or many clusters, where each cluster will represent an hypothesis of the robot pose. Finally, we select the hypothesis with the highest likelihood, provided that it exceeds a minimum threshold min_{bel}. The likelihood of each hypothesis is computed as the sum of the weights of the particles that it contains.

Additionally, this approach lets us detect three situations. First, there is only one cluster with a high likelihood (one cluster with $bel > min_{bel}$), therefore we can infer that the algorithm is self confident about the robot's pose. Second, there are a few clusters with high likelihood (more than one cluster with $bel > min_{bel}$), which tell us that the algorithm is unsure about which one is the robot's pose because there are similarities in the environment. Third, the robot does not yet have an hypothesis about its position (the clusters does not reach min_{bel}), there it should keep moving a bit more until the algorithm is able to converge to a position.

4.4 The sensor's measurement models

There are three different sensors in our robot that we can use for location purposes: a 2D laser scanner, the Wi-Fi, and a magnetic compass. Eq. 4.7 provides us with an elegant way to fuse these sensors: the product of the *measurement models* of each sensor. This way confers great scalability to the system, because it allows us to work with a variable number of sensors. On the one hand, we can add the new sensors to our robot (i.e. installing a new laser scanner on the robot). On the other hand it ensures that our system can work even if only a subset of the sensors is available, which is very important, because sensors usually have different data rates or might even fail during operation (i.e. out of batteries). Precisely, the *measurement models* are a key element of any localization algorithm based on a particle filter, because

their performance has a high impact in the particle weights. In the following subsections we describe the the *measurement models* of each of our sensors.

4.4.1 2D laser scanner

At any time instant t, a 2D laser scanner provides a vector $\vec{l_t} = \{l_t^1, l_t^2, ..., l_t^{N_L}\}$ of N_L range measurements between the robot and the nearby objects, that vector is usually called the laser signature. We can pre-compute the laser signature $l_e(\vec{s})$ expected for any possible pose in this map using an occupancy map of the environment [86]. Thus, we can approximate the *laser's measurement model* by the similarity between $l_e(\vec{s})$ and $\vec{l_t}$:

$$p(z^l|\vec{s}) = \left[\sqrt{1 - \frac{\sqrt{\sum_{i=1}^{N_L} l_t^i \cdot l_e^i(\vec{s})}}{N\sqrt{\sum_{i=1}^{N_L} l_e^i(\vec{s}) \sum_{i=1}^{N_L} l_t^i}}} \right] \left[\frac{1}{N_L} \sum_{i=1}^{N_L} max\left(1 - \frac{|l_e^i(\vec{s}) - l_t^i|}{max_{LD}}, 0\right) \right] \tag{4.8}$$

The first term of the equation measures shape similarity between both laser scans, by means of their Hellinger distance [87], which measures shape similarity amongst them. In order to take scale into account, the second term calculates the average difference amongst each pair of range measurements $(l_t^i, l_e^i(\vec{s}))$, normalised in the range $[0,1]$ using a triangular function. The parameter max_{LD} (maximum laser difference) indicates the maximum allowed difference between each pair of laser ranges.

A requirement for using a laser scanner for positioning is an occupancy map of the environment. The creation of such a map should be done during robot's deployment. That is, as soon as the robot arrives in a new placement (i.e. a museum, event or exposition), an expert has to move the robot for all over the environment. While doing so, the robot will collect the laser signatures and its odometry to build a map. This task is best known as Simultaneous Location And Mapping (SLAM), and among the myriad of techniques to accomplush it, we have chosen to use the ROS wrapper for OpenSlam's Gmapping [17], which is a popular state of the art solution. Depending on the complexity of the environment the map might take from a few minutes to a couple of hours to be ready, therefore this step makes up the bulk of the time that we need to deploy our tour-guide robot in a new place.

4.4.2 Wi-Fi

Over the past years, a very large number of access points (APs) have been deployed in both public and private buildings. All these APs repeatedly broadcast a signal announcing their existence to the surrounding area. This signal can travel up to several hundreds of meters in open spaces. However, indoor environments usually contain many walls and different pieces of furniture that help attenuating the signal, creating many regions with unique finger printings composed of several signal strengths, audible by any Wi-Fi enabled device like the computer of our robot. Those unique fingerprints are very important, because they allow the discrimination between different areas with similar laser signatures.

The strategy that we have followed to create the measurement model for the Wi-Fi sensor, is to first translate Wi-Fi signals into an intermediate estimate of the robot's position $z^w = (x^w, y^w)$, and then obtain the measurement model from these estimations.

Initially, in order to translate the audible Wi-Fi signals to estimates of the robot position, we have used one of the most popular existing commercial solutions, Ekahau [1]. This solution is regarded as one of the most precise available according to several surveys [88, 89], which analysed different commercial Wi-Fi positioning approaches. In the tests that we have done in our research building, we have obtained an accuracy between 2 and 10 metres, despite that Ekahau assures accuracies of up to 1 metre. Unfortunately, the usage of this solution presented a major drawback: an excessive growing of the deployment time of our robot. The Ekahau solution requires us to move all around the environment constantly informing the Ekahau computer of its position, in order to build a training set which consisted of signal strengths associated to map positions. Moreover, we found more minor disadvantages like periods of time where the system did not provide an estimate of the position for several seconds [83].

Later on, we have decided to create our own intermediate Wi-Fi positioning system [84]. The one we have designed overcomes the major drawbacks of Ekahau, providing z^w estimates at a sustained rate of one per second and allowing us to automate the process in which we collect the training data. Now, we are able to collect the Wi-Fi signal strengths at the same time that an expert moves the robot to create the occupancy map for the laser scanner. Moreover, we have created an automated process that using the robot's trajectory during the creation of the map, it is able to assign a position to each set of Wi-Fi signal strengths. In this way, we obtain a training set for the Wi-Fi without increasing the deployment time of our robot.

[1] $http://www.ekahau.com/$

We estimate robot position in world coordinates x^w and y^w using two regression functions f_x and f_y:

$$x^w = f_x(POW_1, POW_2, ..., POW_{N_w}) \tag{4.9}$$

$$y^w = f_y(POW_1, POW_2, ..., POW_{N_w}) \tag{4.10}$$

where N_w is the number of APs in the environment, and POW_i is the power in dBms of the i^{th} AP. We have chosen the ε-Support Vector Regression technique with Gaussian Radial Basis Function kernels (ε-SVR-RBF) [90] to learn f_x and f_y. The prediction error of the ε-SVR-RBF can be approximated by a zero mean Laplace distribution [90], and therefore the *measurement model* of our Wi-Fi sensor is defined by:

$$p(z^w|\vec{s}) = \left[\frac{1}{2\sigma_x^w} e^{-\frac{|x^w - x|}{\sigma_x^w}}\right] \left[\frac{1}{2\sigma_y^w} e^{-\frac{|y^w - y|}{\sigma_y^w}}\right] \tag{4.11}$$

where σ_x^w and σ_y^w are noise parameters estimated experimentally during a pre-operation (or deployment stage) in which the robot learns both f_x and f_y. We have performed an experiment at the CITIUS research centre (Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela, Spain), in an area of $750 m^2$ approximately. The goal of this experiment was to estimate the noise parameters of Eq. 4.11, as well as, to evaluate the accuracy of our Wi-Fi positioning system. Therefore, we have divided our training data into three data sets: one for training with $N_{train} = 1891$ patterns, one for validation with $N_{val} = 722$ patterns, and one for testing with $N_{test} = 644$ patterns. Each pattern contained 31 input values (AP signal strengths) and 2 outputs (x and y coordinates of the measurement position).

Using these data sets, we have trained two ε-SVR-RBF with $\varepsilon = 0.001$, and evaluated different combinations of the remaining parameters [90], C and γ, with the validation set. The best combination was achieved with the values $C = 2^3$ and $\gamma = 2^{-16}$, using these parameters, we have achieved an average error of 4.87 *meters* in the test set, which is similar to the Ekahau solution, but without the drawbacks that we have mentioned earlier. Finally, we have estimated the noise parameters of the *measurement model* through a *5-fold cross validation* with the training and validation sets [90], which yielded the values of $\sigma_x^w = 2.59m$ and $\sigma_y^w = 2.42m$.

4.4.3 Magnetic compass

A magnetic compass is a low cost sensor available in many robots, that provides the orientation θ^c of the robot with respect to a fixed frame. The drawbacks of this sensor is that it is heavily affected by local disturbances of the magnetic field, as those generated by water or electric sources. Nevertheless, in most cases it is able to precisely determine the heading of robots.

Assuming that the measurement noise is Gaussian:

$$p(z^c|\vec{s}) = \left[\frac{1}{\sigma^c\sqrt{2\pi}} e^{-\frac{(\theta_c-\theta)^2}{2(\sigma^c)^2}} \right] \tag{4.12}$$

where θ is the orientation component of robot pose $\vec{s}$, and σ^c is a noise parameter of this sensor, which can be estimated experimentally.

This assumption is reasonable in absence of strong magnetic interferences, which would require a more complex alternative. Nevertheless, as we describe in the next section, the magnetic compass helped our robot in obtaining a better estimate of its pose.

4.5 Evaluating the performance of our multi-sensor localization algorithm

We have carried out several experiments with our localization algorithm. The goal of this experiments is to evaluate the actual benefit of fusing information from many sensors, as well as, to evaluate the performance of the system in order to use it for route recording and reproduction in our tour-guide robot.

In the experiments, we have recorded two different robot trajectories in one floor of our research centre (Fig. 4.1). The first one is a 100 meters trajectory (performed in 161 seconds), that starts in the centre of L_1 and goes trough the main corridor and goes down the ramp to finish in the lowest level, and the second is a trajectory where the robot is just spinning for 51 seconds inside L_1. In order to obtain the ground truth for these trajectories, we initialised our algorithm with the actual pose of the robot and then, each new estimated pose was supervised by an expert to either accept or correct it. After comparing the positions estimated by the robot along the two trajectories with the ground truth, we found a maximum error of 30 cm. We are aware that the use of an external and high precision localization solution would have been a

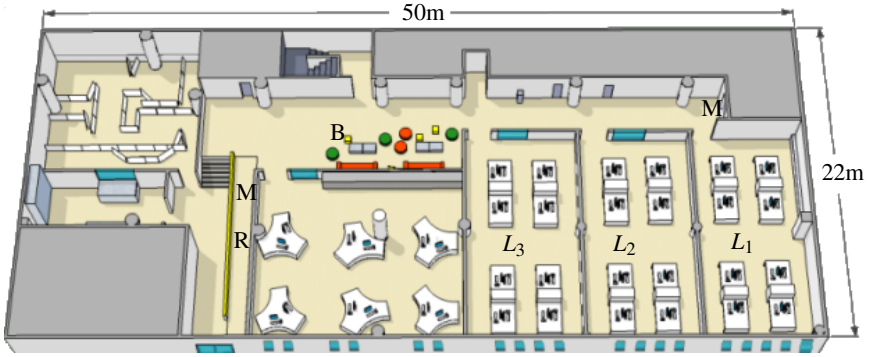


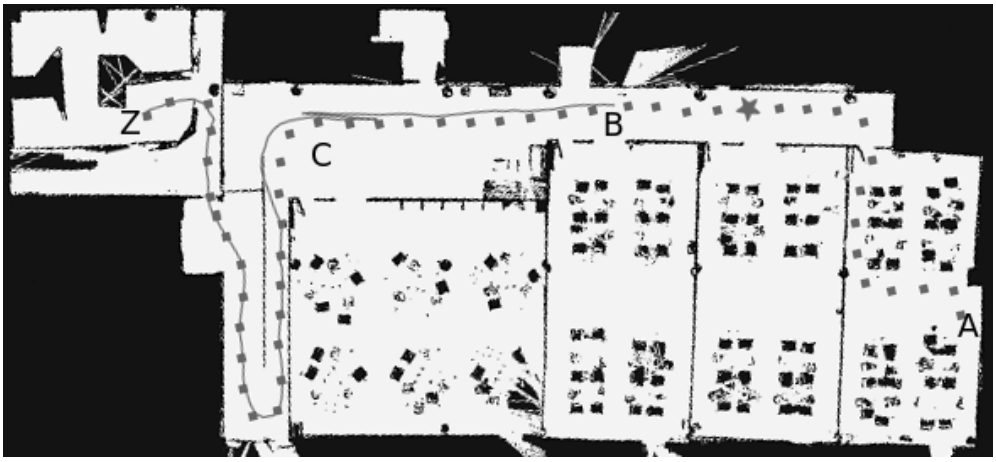
Figure 4.1: Illustration of the underground floor of our research centre where we have tested the localization algorithm. The main challenges are: three laboratories (L_1,L_2,L_3) that are very similar to each other, a large corridor with movable furniture (B), a descent ramp (R), and two points of strong magnetic interference (M).

better option to build the ground truth. However, we did not have access to any solution that is able to work on such a big area.

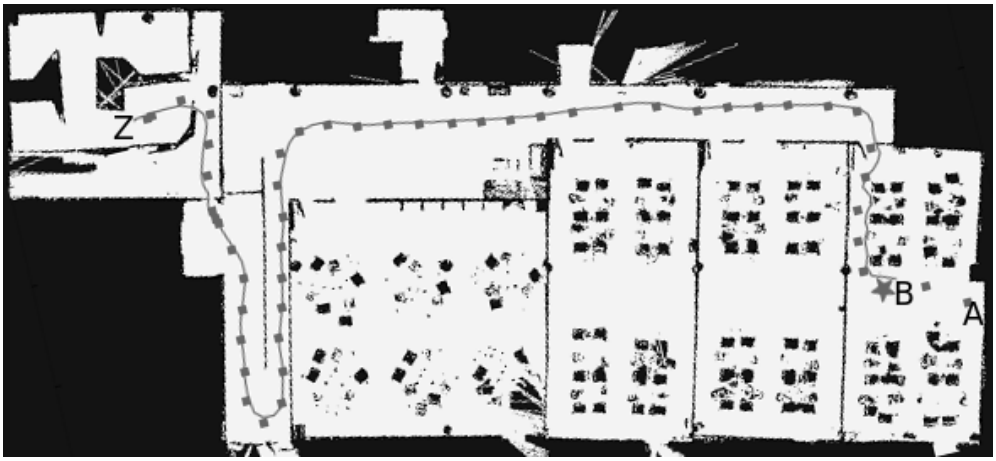
Once data was ready, we have evaluated the localization algorithm with different sensor combinations: 1) laser only (L), 2) laser and compass ($L+C$), 3) Wi-Fi only (W), 4) Wi-Fi and compass ($W+C$), 5) Wi-Fi and laser ($W+L$), and 6) Wi-Fi, laser and compass ($W+L+C$). We ran our algorithm 10 times with each combination, without providing any information about the initial robot pose within the map. Each localization step always took less than 100 ms in the robot's processing unit (Intel Core i7-3610QM). In all the experiments, we used the following parameters: 1) $M = 1000$ particles (Sec. 4.3.1), 2) $max_{CD} = 3\,m$ (Sec. 4.3.2), 3) $min_{bel} = 0.75$ (Sec. 4.3.2), 4) $max_{LD} = 2\,m$ (Sec. 4.4.1), 5) $\sigma_x^w = \sigma_y^w = 5m$ (Sec. 4.4.2), 6) $\sigma^c = 1\,rad$ (Sec. 4.4.3).

The results that we have obtained from these experiments can be seen in tables 4.1 and 4.2, which allow us to extract the following conclusions:

1. *Laser-only (L).* The low $\%t_{loc}$ indicates that the location algorithm needs excessive time to converge to a position (as it is illustrated in Fig. 4.2(a)). The low percentages of $e_{xy} < 5m$ and $e_\theta < \pi/4$ also demonstrate that there is a large probability of incorrect convergence. This happens because the laser alone can not distinguish amongst similar rooms. This causes the algorithm to either not to select an hypotheses, or to select one randomly.



(a) Poor localization results when using only laser data.



(b) High localization performance example when using all the sensors in our robot (laser, wifi and compass).

Figure 4.2: Two trajectory samples (grey line) generated with our localization algorithm. The ground truth trajectory is represented with grey squares. Letters A and Z represent the start and end points, respectively, of the trajectory. Letter B represents the point where our algorithm starts to output a pose estimate, and the grey star represents the actual position of the robot at that moment. Letter C indicates a zone in Fig. 4.2(a) where our algorithm had to correct its pose estimate.

Traj.1	L		$L+C$		W		$W+C$		$W+L$		$W+L+C$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
e_{xy} (m)	1.81	2.25	2.67	2.88	2.72	0.30	2.54	0.36	0.77	0.38	**0.52**	0.15
e_θ (rad)	0.14	0.18	0.07	0.02	0.25	0.05	0.12	0.03	0.15	0.12	**0.08**	0.02
$e_{xy} < 5m$ (%)	87.5	18.2	76.5	23.0	93.4	2.37	98.6	0.07	97.6	3.45	**99.3**	0.00
$e_\theta < \pi/4$ (%)	97.8	5.80	100	0	95.0	3.50	98.3	0.67	97.2	0.45	**99.8**	0.06
t_{loc} (%)	74.4	15.6	79.9	7.7	76.6	6.4	87.0	3.68	84.2	2.63	**93.3**	2.18

Table 4.1: Results for trajectory 1. We provide averages (μ) and standard deviations (σ) of: error in position (e_{xy}) and orientation (e_θ), percentage of time that the error in position is below 5 meters, percentage of time that the error in orientation is below $\pi/4$ *rad*, and percentage of time that the algorithm is able to provide a localization estimation (t_{loc}). Bolded numbers highlight the best results.

Traj.2	L		$L+C$		W		$W+C$		$W+L$		$W+L+C$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
e_{xy} (m)	**0.42**	0.44	5.65	5.77	1.22	0.42	1.83	0.71	0.92	0.61	0.59	0.30
e_θ (rad)	0.14	1.68	0.07	0.05	1.80	0.57	0.17	0.06	1.62	1.37	**0.18**	0.11
$e_{xy} < 5m$ (%)	100	0	44.4	52.7	100	0	99.0	2.23	100	0	**100**	0
$e_\theta < \pi/4$ (%)	66.7	57.7	100	0	13.4	21.5	100	0	48.9	48.4	**100**	0
t_{loc} (%)	13.9	25.2	41.0	25.3	45.1	15.9	70.2	18.6	60.4	13.6	**80.6**	4.59

Table 4.2: Results for trajectory 2. We provide averages (μ) and standard deviations (σ) of: error in position (e_{xy}) and orientation (e_θ), percentage of time that the error in position is below 5 meters, percentage of time that the error in orientation is below $\pi/4$ *rad*, and percentage of time that the algorithm is able to provide a localization estimation (t_{loc}). Bolded numbers highlight the best results.

2. *Laser and compass (L+C).* The compass deteriorates the performance of the laser: it tends to reduce the convergence time, at the cost of increasing e_{xy}. Again, none of the sensors can discriminate amongst similar rooms, but the compass information tends to reduce the diversity of particles. This increases the probability of convergence towards a random hypothesis.

3. *Wi-Fi only (W).* It has high e_{xy} values, but the percentage of $e_{xy} < 5m$ is also quite high. This means that even though Wi-Fi might not have a high accuracy in short distances it is very adequate to discriminate amongst spatially distant hypotheses, for instance, amongst the laboratories (L_1,L_2,L_3) or the part of a large corridor like the one in our building.

4. *Wi-Fi and compass (W+C).* The compass improves slightly the performance of the Wi-Fi. Noticeably, it reduces the percentage of $e_\theta > \pi/4$ to a 0%.

5. *Wi-Fi and laser (W+L).* The Wi-Fi helps the laser to discard spatially distant hypotheses. This improves $\%t_{loc}$, and increases the percentage of $e_{xy} < 5$. However, the Wi-Fi adds noise, which tends to increase the values of e_{xy} and e_θ.

6. *Wi-Fi, laser, and compass (W+L+C).* This is the best combination, as we illustrate in Fig. 4.2(b), this option overcomes the drawbacks shown in Fig. 4.2(a). It shows a percentage of $e_{xy} < 5$ above the 99%, and a percentage of $e_\theta < \pi/4$ of 100%. Moreover, the high value of t_{loc} (80.59% in trajectory 2 and 93.3% in trajectory 1), and the low σ values indicate a fast and stable convergence (convergence time below 11 seconds).

7. *Trajectory 1 vs 2.* If we compare the results obtained for the two trajectories, we can see that the main difference is that $\%t_{loc}$ is much higher in the first one. This happens because in the second trajectory the robot is just spinning, continuously obtaining the same (or similar) sensor data, therefore the data is not discriminative enough to obtain a fast first estimate of the robot's pose.

To summarise, we have proved that the combination of information sources with non-correlated errors tend to perform better than solutions based on single sensors alone [83]. More specifically, the combination of laser, compass and a Wi-Fi positioning system can help a tour-guide robot to obtain a robust and stable estimate of its position. It avoids the confusion between similar spaces while it provides accurate localization estimates. Moreover, our solution can handle a variable number of sensors (scalability), even if they have different data rates.

4.6 The route recording architecture

An instructor may launch the route recording process using the AR-GUI (chapter 3) at any time during the operational stage. The process assumes that during the entire recording, the estimation of the pose is correct (error below 1 meter). This is the reason why we have made an special effort into obtaining a robust localization solution.

Figure 4.3 shows the scheme of the route recording architecture. The location module is an asynchronously, always-working, independent node. That is, it is continuously estimating the robot's pose with every new sensor information available. In the case of the route recording process, it is launched by the instructor, and then it remains in the background, waking up in

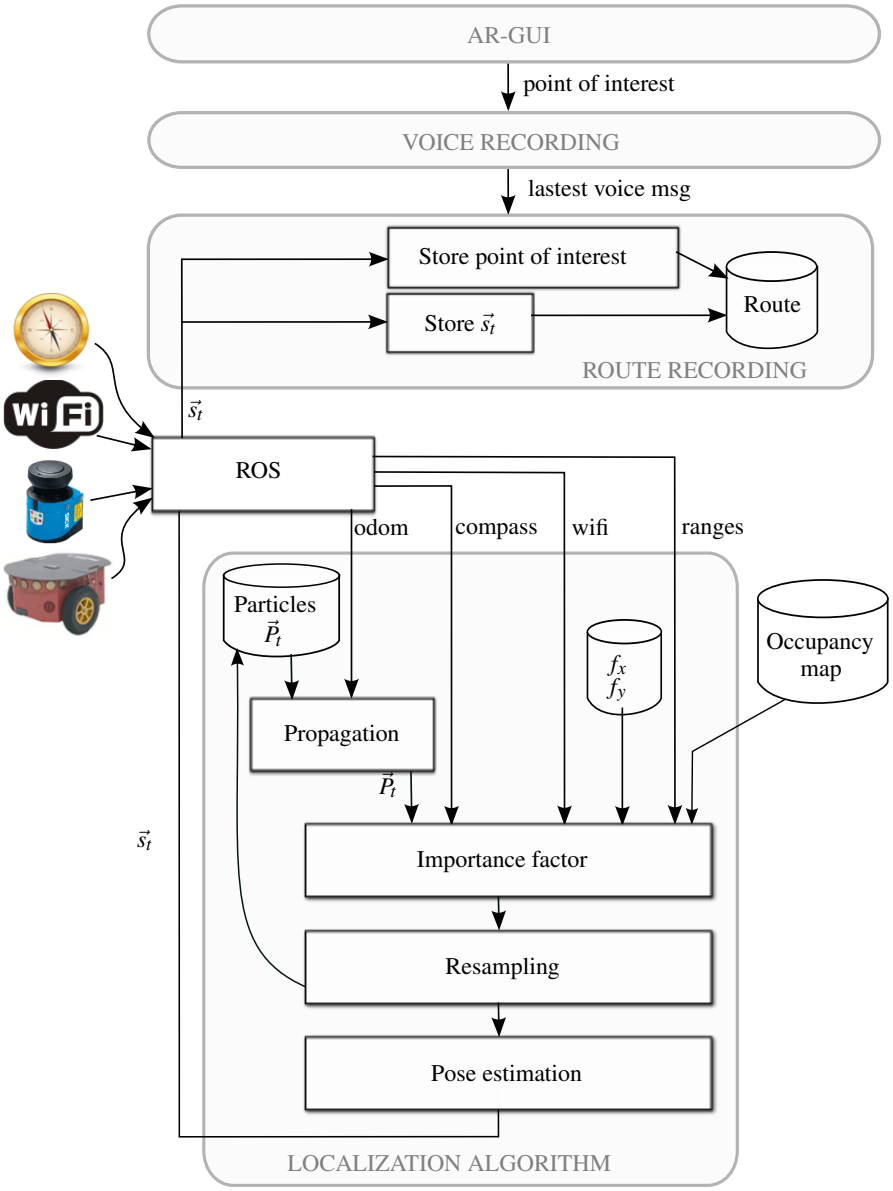


Figure 4.3: Route recording architecture.

two cases: a) every time that the location algorithm estimates a new pose ($\vec{s_t}$), and b) when the AR-GUI informs about a *point of interest*:

- In the first case, it will decide whether to store $\vec{s_t}$ or not based on how far $\vec{s_t}$ is from the latest pose stored in the route.
- In the second case, it will store the voice message recorded by the instructor, alongside the latest estimated robot pose ($\vec{s_t}$).

In this way, the routes of our robot are composed of a sequence of robot poses (which we call *way-points*), intercalated with *point of interest* (which consist of a robot's pose and a voice message). This data will be used by the route reproduction process to mimic the path of the robot and to reproduce the voice messages at the same places where they were originally recorded.

4.7 Conclusions

In our tour-guide robot, routes are dynamically learnt from humans who act as instructors. The interaction between the instructor and the robot has already been described in the previous chapters, while on this chapter we focus on the actual process of route recording. This process is mainly based on the recording of a sequence of robot's poses.

Our main contribution can be found in the description of a multi-sensor localization algorithm that we have developed. Regarding it, real world experiments have shown that the localization algorithm is accurate, robust, and fast, and that it can take advantage of the strengths of each sensor minimising their weaknesses. These advantages make it a good solution for positioning during the route recording task.

Moreover, we have described the architecture of our solution to gather route information, as well as its interaction with the rest of the elements of our robot. The main advantage of our solution for route learning is that it is not performed on the deployment stage, which would require the intervention of an expert and increase the robot's deployment time. In this way, the tour-guide robot is deployed in a matter of very few hours, and then it is ready to start learning new routes and to start showcasing the ones that have already been learnt.

Nevertheless, in the future, the suitability of the localization algorithm should be extensively tested in crowded environments where many people might surround the robot to interact

with it. Such crowded environments might reduce the performance of the solution, and could eventually prevent the robot from correctly recording or reproducing a route.

CHAPTER 5

ROUTE REPRODUCTION

A visitor of an event can demand a route from our tour-guide robot in an easy way, as we have described in chapter 3. In this chapter, we describe what happens in the robot after the visitor has selected that route.

Before our tour-guide robot starts the reproduction a previously recorded route, it needs to travel to the starting point of the route. This will require to plan and execute a valid path from its position to that point. After that, the planning and execution of a path will be an easier task, because the route is composed by a series of *way-points* which are not far from each other. Every time that our robot reaches one of those *way-points* it will make short *beep* sound, to let the user's know that the route is being reproduced correctly. Additionally, whenever it reaches a *point of interest* it will reproduce the associated voice message and then it will keep moving towards the next *way-point* in the route. Finally, when the robot has finished showing the route he will double *beep* and keep waiting for any command from the visitors of the event.

The major challenge that we address in this chapter is the software architecture of the route reproduction behaviour. This architecture includes finding an appropriate solution for path planning, robot navigation and dynamic obstacle avoidance. The goal is for the robot to safely navigate from way-point to way-point within the route path, avoiding collisions with the environment or any dynamic obstacle that might appear, such as humans.

5.1 State of the art

Path planning has been studied in many fields in which it is necessary to traverse a graph. The robotics community applied results from these studies to fit its own problem. In this

regard Dijkstra's algorithm [91] and the A* search algorithm [92] are the two most popular algorithms currently used for path planning. However, there are many others which has also been quite popular in the latest years, such as Anytime Dynamic A*(AD*) [93], Rapidly-exploring Random Trees (RRT) [94], any-angle path planning on grids (Theta*) [95], ant colony optimisation algorithms [96] or genetic algorithms [97].

Robot navigation has been extensively studied over the past years as it is a key task of any mobile robot. In this regard, the most popular strategy is the dynamic window approach [98], which is able to avoid dynamic obstacles while seeking a goal position. It consists in forward simulating the robot position with different angular and linear velocities. The obtained positions are then scored based on their proximity to obstacles and to the goal position. Therefore, the velocities with the highest score are chosen and executed in the robot. The dynamic window approach was presented with Rhino [2, 3] which is the first well-known tour-guide robot, and then it has been used in most tour-guide robots over the past fifteen years [5][6].

There are many other approaches to robot navigation and obstacle avoidance, which are also popular mainly due to its simplicity. One of these popular approach is the classic robot controller based on potential fields [37]. In this kind of controllers there are two types of forces, an attractive force which points to the robot's goal and several repulsive forces which are produced by the obstacles. The sum of the forces should give the direction that the robot should follow to avoid obstacles. We have used this controller in early versions of our tour-guide robot [19]. In fact, given that it is very simple and requires a very low computational cost, we are still using it to follow a human while learning a route.

5.2 Route reproduction architecture

Event attendees can demand any route from our tour-guide robot, regardless of the place where the robot is. This means that the first task that our robot has to accomplish is to travel to first point of the route from its current position. Then, it will need to reproduce the route in the most accurate possible manner, dealing with the dynamic nature of the environment.

In order to describe this process, we can define a route r as a list of robot's poses $\{\vec{s_1}, ..., \vec{s_w}\}$ where w is the number of stored poses (or *way points*). Some of these *way points* can actually be *points of interest*, if they are associated with voice messages. The goal of the route reproduction behaviour of our robot is to visit each of those points in the specified order, and to reproduce the associated voice message if there is one.

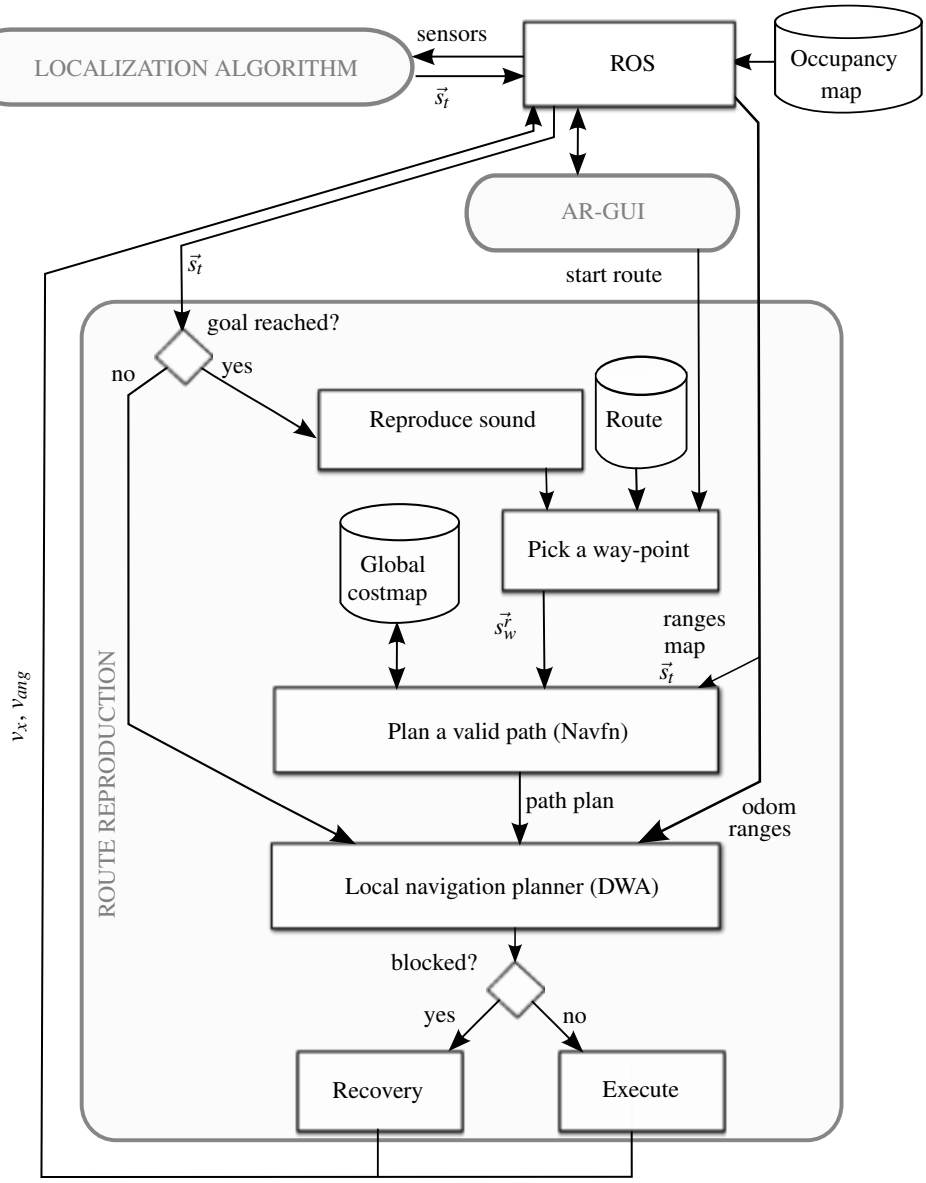


Figure 5.1: Schematic representation of the navigation node used for route reproduction in our tour-guide robot. It is a ROS based node with a global planner for path planning (navfn) and a local planner based on the Dynamic Window Approach (DWA) for navigation.

In Fig. 5.1 we can see that the route reproduction process starts when the AR-GUI demands a route (*start route*). At this moment, the behaviour picks a *way point* ($\vec{s}_1^r$) and plans a valid path from the robot's pose ($\vec{s}_t$) to that point. These points can be separated several dozen of meters, and planning a valid path might not be a trivial problem. We have chosen a classical solution, Dijkstra's algorithm [91] to tackle this problem. It is a good solution in terms of efficiency and simplicity, and it was able to provide a viable global plan in all cases that we have tested. More specifically, we have used an improved version which is available in the ROS package *navfn* [1]. This algorithm requires that we build a cost-map of the environment where the robot operates (*global costmap* in Fig. 5.1). This costmap will be used as the graph in which Dijkstra's algorithm will search for a minimum cost plan between the two points. In order to build the costmap of the environment we need: an occupancy map of the environment (which is created during the deployment stage of the robot, as described in section 4.4.1), the robot's estimated pose $\vec{s}_t$, the current data from the laser scanner, and the radius of the robot min_r.

The costmap is initialised using the occupancy map of the environment: we assign the highest value in the costmap to those cells that are marked as occupied in the occupancy map, and a low value to those cells that are marked as free space. Then, using the robot's estimated pose and the laser signature, we create new occupied cells in the costmap, and in the same way, we also release cells that were previously marked as occupied. We need to do this because there might be obstacles in the environment that are not in the map, such as humans, but there might also be pieces of furniture that changed their place, releasing previously occupied space. The last step to build the costmap consist in marking as occupied those free cells that are closer than min_r, as the robot will not be able to visit those cells without colliding. This process is known as *inflating the obstacles* of the map.

Our algorithm will use the costmap to plan a valid path from the robot's position towards the start point of the route. This plan will be handed out to a second element: the local navigation planner (Fig. 5.1). The local planner is the element responsible for safely moving our robot in the environment, while it follows the path plan. In order to face this problem, we use a popular algorithm that is able to solve the task with an small computational overhead: the Dynamic Window Approach (DWA) [2] [98]. Essentially, this algorithm forward-simulates various trajectories which are generated by sampling the linear and angular velocities of the

[1] `http://wiki.ros.org/navfn`

[2] `http://wiki.ros.org/dwa_local_planner`

robot. Then, each simulated trajectory is scored according to different criteria. In the end, the trajectory with the highest score is chosen, and the velocities which generated that trajectory are executed in the robot's base. This algorithm, allows us to continuously re-evaluate the robot's trajectory, and thus avoid dynamic obstacles such as persons walking in the robot's path.

An important element of the dynamic window approach is how we score the simulated trajectories. For this, in each cycle we need to create a second costmap, which we call the local costmap. In order to do this, we use the laser signature and the computed path plan. First, obstacles are inflated in the same way than in the other costmap, and then we assign different weights to free space cells, according to how far they are from the planned path. Therefore, using this local costmap, trajectories can be scored taking into account how close they are to the desired path and on different parameters like proximity to obstacles or achieved speed.

When our tour-guide robot has reached the start point of the route, it will make a short beep (*Reproduce sound* in Fig. 5.1), pick the next *way point* and start a similar process to navigate towards it. That is, we will compute a new path plan using Dijkstra's algorithm and safely execute it using the Dynamic Window Approach. We repeat this process until the last point of the route is reached or the visitor stops the robot. Whether or not it is necessary to compute a new plan to the next *way point* will depend on how far is that point from the current robot's position. That is, in the case that the next *way point* is a few centimetres away, we could skip that step. Additionally, our route might contain *points of interest*, which are treated similarly to a normal way-point. The main difference is that the robot will stop in that point for a while, until the associated voice message is fully played back (*Reproduce sound* in Fig. 5.1).

Finally, we have also implemented several rules to increase the flexibility of the robot, and to deal with unreachable route points. First, we consider a point as visited when the robot moves at a close distance (0.5m from the route point). This prevents our robot from spending many seconds moving slowly to reach a goal pose with a high precision, thus increasing the overall robot's speed. Second, when our robot's detect that a point within the route is unreachable, e.g. a human is blocking the path, we can proceed in two different ways. In the first one, the robot will clear its local costmap from the nearest obstacles and perform an in-place rotation to check for alternative paths. If there is still no valid path, the robot will perform a more aggressive reset of the costmap, completely clearing it and performing another

in-place rotation to rebuild it. If this also fails, we will consider the second alternative: the robot is completely blocked, and therefore we will allow it to skip that *way-point* and move to the next one. In case that the robot skips a point, it will make a recognisable error sound to let the user's know that there were a problem.

5.3 Tests in the robotics laboratory

Figure 5.2: $100m^2$ robotics laboratory where we have conducted several route reproduction tests. We set up an environment with 5 rooms, and recorded and reproduced several routes while two humans were walking around.

In this section, we want to illustrate the performance of the route reproduction module that we have designed for our tour-guide robot. For this reason, we will describe two routes that we have recorded in our robotics laboratory (Fig. 5.2). In this laboratory we have settled an indoor environment with five rooms.

The routes that we have recorded are illustrated in Fig. 5.3 with a dotted line (robot's trajectory during recording) and small dark grey circles (recorded way-points). The result obtained when reproducing the route is illustrated in Fig. 5.3 with a light grey line. In these examples we can see how the robot has to previously find a valid path from its initial pose (C) to the initial point of the route (A). Once the robot has achieved the first point of the route, it moves from one *way-point* to the next one, avoiding obstacles in the environment (two humans walking randomly) without separating from the path more than a few dozens of

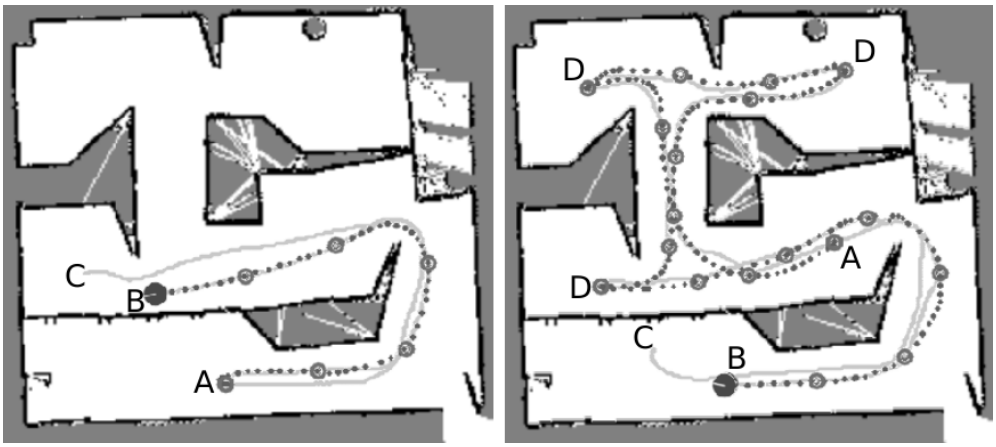


Figure 5.3: Robot trajectories while recording (dotted) and reproducing (light grey) two routes. The small circles represent the stored way-points. The points marked with the letters *A* and *B* represent the first and last points of each route, respectively. The points marked with letter *C* are the location of the robot when it was asked to reproduce the route, and the points marked with the letter *D* in the second route are the locations of *points of interest* as specified by the route teacher.

centimetres. Moreover, when the robot reached a point of interest, it played the voice message recorded by the instructor. Finally, when the robot finished the reproduction of the route, it played a bell sound and waited for new commands from any other human.

The first route (Fig. 5.3, left) was recorded in 56 seconds, and it is 15.3m long. In this route no points of interest were recorded by the instructor, but it is interesting in order to notice how fast can anyone teach a route path to the robot in a home-like environment with narrow corridors. In this experiment, the robot had to travel 29.2m in order to find the first point of the route and mimic the complete route path.

The second route (Fig. 5.3, right) was recorded in 155 seconds, and it is 40.8m long. When reproducing this route the robot travelled 52.8metres. In this route, the instructor recorded three voice messages at the points of interest. These messages were very short: around 5 seconds each one. Our tour-guide robot allows the recording and the reproduction of a complex route in little time by non experts.

Moreover, we have to remember that we have showcased our tour-guide robot during many demonstrations in our research centre (CITIUS). These demonstration are part of a program for visitors, in which we show them the results of our research activity. Fig. 5.4 shows a picture of a visitor teaching our robot a route inside our laboratory (Fig. 5.2). Then,

Figure 5.4: Pictures of several demonstrations in which we have showcased the route recording and reproduction behaviours of our tour-guide robot. In the top picture the human is teaching a route to the robot, in the rest of the pictures the robot is reproducing it.

the rest of the pictures show how our robots reproduces a route that it has just learnt. In many demonstrations our tour-guide robot did not have problems to reproduce the routes, however in some of them it had to perform in-place rotations to clear out blocked space. This situation arises when the environment differs significantly from the one in the occupancy map, which might be a common problem when the visitors are mainly kids who play around with furniture in our laboratory.

5.4 Conclusions

The route reproduction behaviour that we have designed and implemented in our tour-guide robot is able to mimic a route that has been previously learn.

We have used popular solutions for path planning, obstacle avoidance and robot navigation, which allow our robot to safely move in a dynamic environment to reproduce the route. The robot is able to search for the start of the route regardless of its position when the visitor demands a route. Moreover, we have also implemented several solutions to tackleblocking paths or other problems that the robot might encounter during the reproduction of a route. Through many tests with visitors of our research centre, we have demonstrated the feasibility of this task in our robot.

In the future, we believe that the robot should include a more complex behaviour than the current one, which should include a solution to be aware of what the person that demanded the route is doing. That is, the robot should wait for that person when showcasing a route and detect if the person has lost interest in that route and stop reproducing it.

The route reproduction behaviour that we have designed and implemented in our tour-guide robot is a modular piece of software that integrates with ROS. This behaviour only requires a robust localization algorithm working on the robot, and a route path (a list of robot's poses) to work in any ROS robot. Moreover, the route reproduction behaviour of our robot should be tested in a museum or at a big and crowded event.

Chapter 6

Learning of the robot controller

The main goal of the proposed thesis is to build a general-purpose guide robot which is able to learn routes from people and repeat them at any moment even in crowed and unstructured environments. Nevertheless, besides the achievement of this kind of guide robot, we also want that the research done in this thesis might help to get closer to personal service robots able to get along with people and perform tasks that directly influence on their living. In this sense, in this thesis we always wanted to obtain algorithms and behaviours capable of self adaptation, like the visual descriptor that maximizes the differences between people (distractors) and the target. However the robot controller that is responsible for deciding the movement of the robot when it is following a person has been developed ad-hoc. In all the experiments we showed so far, we have used a pre-installed controller that determines the linear and angular velocity of the robot considering the obstacles in the environment (repulsive forces) and the target being followed (attractive force).

If we consider the main scenarios where future robots are expected to move, or the tasks they are expected to carry out (assisting with the housework, security and vigilance, rehabilitation, collaborating in the care-entertainment, etc), we immediately realize that this new generation of robots must be able to learn on their own. They can not rely on an expert programmer, on the contrary, once they are bought they should be âtrainableâ. Nevertheless, this learning or robot-adaptation can not only consist on a demonstration process, in which the user shows the robot what to do. The limited nature of the human patience, the ambiguous nature of the information provided by the robot owners, the advanced years or impaired mobility of the robot owners, deem it necessary that not only should robots be able to learn from what

the user does, but also from their interaction with the physical and social environment. Like humans, robots should be able to learn from their own experiences when emulating people or exploring an environment. The mistakes and successes the robot makes should influence its future behaviour rather than relying only on predefined rules, models or hard-wire controllers. This will result in robots that are able to adapt and change according to the environment. No pre-defined knowledge will be used to control the robot. Most of the robotâs competences will be learned through direct physical interaction with the environment and human observation.

For this reason we want to go a step further and provide our robot with the ability to learn its own controller that is able to follow a person while avoiding obstacles of the environment. Moreover, given the limited patience of a human, our goal is also to design an algorithm that is able to achieve fast learning processes. Our proposal is based on the reinforcement learning paradigm so that anyone can tell the robot when it is doing wrong using a wireless joystick. This way of providing reinforcement allows us to avoid the intervention of an expert in robotics in new environments and at the same time we allow the robot to learn from its own experiences and therefore adapt itself to the environment in which it is working.

In this chapter we briefly describe the current robot controller which is based on the potential fields method, and then we describe the proposed solution the allows anyone to teach their own adaptive robot controller.

6.1 The potential field controller

In this section we describe the ad-hoc controller that we have designed and used in our robot. The goals of this controller are: a) to avoid obstacles and b) to maintain the human at a safe distance, centred in the robotâs field of view.

In our robot the safe distance d_s is established at 1 meter. The ad-hoc controller that we have designed is based on the potentials field technique [37], in which the target position exerts an attractive force, while the obstacles around the robot determine the repulsive forces.

More specifically, to determine the repulsive vector, we use the n measurements in the laser signature according to Eq. 6.1:

$$\vec{f}_r = \sum_{i=0}^{i=n} \vec{f}_r^{\,i} \tag{6.1}$$

where $\vec{f}_r^{\,i}$ are the repulsive vectors created from each laser measurement according to the following rules:

– If the measured distance is greater than 1m: $\vec{f}_r^i$ is the unit vector whose origin is the robot's position pointing to the opposite direction of the laser point.

– If the measured distance is lower than 1m: $\vec{f}_r^i$ is the same vector than in the previous but $||\vec{f}_r^i|| = 2 - dist\_to\_obstacle$.

These rules will allow us to modify the robot's trajectory when it travels close to obstacles and ignore them when they are not a danger for the robot.

Similarly to $\vec{f}_r$, the attractive force $\vec{f}_a$ is the unit vector whose origin is the robot's position and points to the human's position.

In the end, we sum both $\vec{f}_r$ and $\vec{f}_a$, and compute the module and orientation of the resulting vector. The module will be the linear velocity that we send to the robot, while the vector orientation will be orientation goal that we will try to reach with the angular velocity that we send to the robot's base, which is computed heuristically. Given that we also aim to maintain a safe distance with the human, we will stop the robot when the target is closer than 1 meter (zero linear velocity).

This method allows us to determine linear and angular velocities that the robot must carry out so that it safely follows the target. However, this method has many disadvantages that are revealed when it is tested in real world scenarios. For example, when there are many people surrounding the robot the repulsive forces are strong and might modify the robot's trajectory in a way that make the robot lose its target. A similar problem is observed when the robot is working in narrow spaces or when it tries to cross some doors: either it moves really slow or is unable to follow its target.

Finally, as we have already stated in the introduction a clear disadvantage of this type of controllers is the time that a programmer might invest in fine tuning the algorithm to make it work at certain environments. Therefore, we believe that our robot would highly benefit from the ability of learning new controllers from the robot's instructor. In this regards, the next section contain the description of our algorithm.

6.2 The learning algorithm

As we mentioned at the beginning of this chapter, instead of working with a pre-installed control software, in this thesis we also want to do research on the topic of learning a robot controller from scratch and on the real robot (not only in simulated scenarios).

One of the most suitable learning paradigms to get a robot learning from its interaction with the environment is reinforcement learning (RL). What makes this learning paradigm appealing is that the system learns on its own, through trial and error, relying only on its own experiences and a feedback â reward signal – that encourages or discourages the execution of different sequences of actions. The reward signal tells the robot how well or badly it has performed, but nothing about the actions it should have carried out. Through stochastic exploration of the environment the robot must find a control policy which maximises the expected total reward it will receive. There are, however, known drawbacks to the application of RL in robotics. Thus the large numbers of random actions taken by the robot, especially during the early stages, or the very long convergence times showed by most of the traditional algorithms, make the application of this learning paradigm in real robots very difficult. In our case we have investigated a new proposal that combines some of the results we have already achieved in our research group with new developments carried out in this thesis, to achieve an algorithm able to reach fast learning process on a real robot interacting in the environment, even when the action space is multidimensional.

6.2.1 Control policies

Let us say that there is a control policy π that determines what the robot does at every instant, i.e., this policy π is a mapping from relevant and distinguishable states to actions:

$$\begin{array}{lll} \pi : S \times A & \rightarrow & [0,1] \\ (s,a) & \rightarrow & \pi(s,a) \end{array} \tag{6.2}$$

where S is the set of states that represent the environment around the robot, and A is the set of possible actions the robot can carry out, and $\pi(s,a)$ is the probability of performing action a in state s.

The goal of reinforcement learning is to discover an optimal policy π^* that maps states to actions so as to maximize the expected return J the robot will receive:

$$J = E\{\sum_{t=0}^{\infty} \gamma^t r_t\}$$

where r_t is the reinforcement received at time t, anf $\gamma \in [0,1]$ is a discount factor which adjusts the relative significance of long-term rewards versus short-term ones.

Therefore, when reinforcement learning is applied in robotics, all a robot needs is a reinforcement function which tells the robot how well or how poorly it has performed, but

nothing about the set of actions it should have carried out. Through a stochastic exploration of the environment, the robot must find a control policy which maximises the expected total return described above. To get information about the rewards and the behaviour of the system, the agent needs to explore by considering previously unused actions or actions it is uncertain about. It needs to decided whether to play it safe and stick to well known actions with (moderately) high rewards, or dare trying new things in order to discover new strategies with an even higher reward [99, 100]. This problem is commonly known as the exploration-exploitation trade-off, and usually involves the development of an explicit exploration strategy which usually makes the robot select actions more randomly at the beginning of the learning process and, on the contrary, be more conservative and pick less new or unexplored actions as the learning process goes by.

There is a wide range of algorithms within the reinforcement learning paradigm and there are many ways of classifying all of them: *On-policy or Off-policy methods*; *value function-based approaches* or *policy search* methods, etc [99, 100]. It is impossible to describe all of them in detail in this chapter. Nevertheless, most of the classical strategies are too slow and involves many random robot actions before it reaches a successful control policy. Since experience on a real robot is tedious to obtain, expensive and often hard to reproduce, quite often reinforcement learning algorithms are applied on a robot-simulator and once the learning process is over, the optimal control policy is placed on the real robot. However, although real-world experience is costly, it cannot be replaced by learning in simulations alone. Because of this, in this thesis we have developed and tested a new strategy which tries to achieve fast learning processes on a real robot which is learning from scratch in a real environment, using a reinforcement signal provided from an human observer, and without the necessity of using an explicit exploration strategy. As we will see in the next sections, our approach starts from previous results obtained in our research group [101, 102, 103] to achieve a proposal that combines: a) a new value function that approximates the expected time interval before the robot makes a mistake (Tbf), b)a novel representation of the action space, that will allow a progressive estimation of the best action for every state. The robot will discover first the best action interval for every state, nevertheless, this interval will be narrower and narrower as the learning process converges. c) a reduction of the learning time and an improvement of the quality of the final control policy thanks to the use of a ensemble of learners

6.2.2 Value functions: evaluating a control policy

To evaluate a control policy, i.e. to quantify how long a policy will be able to move the robot before it makes something wrong and the robot receives negative reinforcement, we will use an algorithm that we have published in the past and which is called *increasing the time interval before a robot failure* [101]. Using this algorithm our system will not learn the expected discount reward the robot will receive – as it is habitual in reinforcement learning–, rather the expected time before failure (*Tbf*). This will make it easier to assess the evolution of the learning process as a high discrepancy between the time interval before failure predicted and what is actually observed on the real robot is a clear sign of an erroneous learning.

To assess our control policy we will build a utility function of the states the robot might encounter, termed *V-function*. Thus, $V^{\pi}(s)$ is a function of the expected time interval before a robot failure when the robot starts moving in s, performs the action determined by the policy for that state $\pi(s)$, and follows the same policy π thereafter:

$$V^{\pi}(s) = E[-e^{(-Tbf^{\pi}(s_0=s)/50T)}], \tag{6.3}$$

where $Tbf^{\pi}(s_0)$ represents the expected time interval (in seconds) before the robot does something wrong, when it performs action $\pi(s)$ in s, and then follows the control policy π. T is the control period of the robot (expressed in seconds). The term $-e^{-Tbf/50T}$ in Eq. 6.3 is a continuous function that takes values in the interval $[-1,0]$, and varies smoothly as the expected time before failure increases.

Therefore, according to Eq. 6.4, if the robot performs an action every 250 milliseconds (value of T in Eq. 6.4), a V-value equal to -0.8 (for example) will clearly mean that the robot will probably receive a negative reinforcement after 2.8 seconds.

$$Tbf^{\pi}(s) = -50 * T * Ln(-V^{\pi}(s)), \tag{6.4}$$

The definition of $V^{\pi}(s)$, Tbf^{π}, determine the relationship between consecutive states:

$$Tbf^{\pi}(s_t) = \begin{cases} T & \text{if } r_t < 0 \\ T + Tbf^{\pi}(s_{t+1}) & \text{otherwise} \end{cases} \tag{6.5}$$

r_t is the reinforcement the robot receives when it follows π in state s_t. If we combine Eq. 6.3, 6.4, and 6.5, it is true to say that in the general case:

$$V^{\pi}(s_t) = -e^{-1/50} \prod_{a} \prod_{s'} (-V^{\pi}(s'))^{\pi(s_t,a)P(s_t,a,s')}. \tag{6.6}$$

where $P(s_t, a, s')$ represents the probability of moving from state s_t to state s' when the robot performs action a in s.

Instead of the value function $V^{\pi}(s)$ many algorithms rely on the *state-action value function* $Q^{\pi}(s,a)$ instead, which has advantages for determining the optimal policy. This new function is defined as the expected time before a robot failure when the robot starts from s, by executing action a, and then following policy π:

$$Q^{\pi}(s,a) = E[-e^{(-Tbf^{\pi}(s_0=s,a_0=a)/50T)}], \tag{6.7}$$

hence:

$$Tbf^{\pi}(s,a) = -50 * T * Ln(-Q^{\pi}(s,a)),$$

Like we did for the V-function, in the case of the Q-values it is possible to say that in general:

$$Q^{\pi}(s_t,a) = -e^{-1/50} \prod_{s'} (-V^{\pi}(s'))^{P(s_t,a,s')}. \tag{6.8}$$

In contrast to the value function (V-values), the state-action value function (Q-values) explicitly contains the information about the effects of a particular action. It is important to be aware of the fact that, considering a particular policy, there will be only one value function for every possible state while, on the contrary, for the same state there will as many state-action values as the number of actions that is possible to perform in that state.

6.2.3 Optimality of policies: lookup table

For any two policies, π and π', we say that policy π' is an improvement over policy π, if $V^{\pi'}(s) \geq V^{\pi}(s)$, $\forall s \in S$, and the inequality is strict for at least one state. A policy is an *optimal policy* if no policy is an improvement over it, i.e., a policy is optimal if it maximizes the value of each state (*optimal value function*). The symbols π^*, V^*, and Q^* are used to represent an optimal policy and the optimal value and action-value functions, respectively. Therefore:

$$V^*(s_t) = -e^{-1/50} * max_a \{ \prod_{s'} (-V^{\pi}(s'))^{P(s_t,a,s')} \} \tag{6.9}$$

and:

$$Q^*(s_t,a) = -e^{-1/50} \prod_{s'} (-V^*(s'))^{P(s_t,a,s')} \tag{6.10}$$

or equivalently:

$$Q^*(s_t,a) = -e^{-1/50}\prod_{s'}(-max_a Q^*(s',a))^{P(s_t,a,s')}. \tag{6.11}$$

It can be shown that an optimal, deterministic policy $\pi^*(s)$ can be reconstructed by always picking the action a^* in the current state that leads to the sate s with the highest value $V^*(s)$:

$$\pi^*(s) = argmax_a(Q^*(s,a)) \tag{6.12}$$

$$\pi^*(s) = argmax_a\{(-e^{-1/50}\prod_{s'}(-V^*(s'))^{P(s,a,s')}\} \tag{6.13}$$

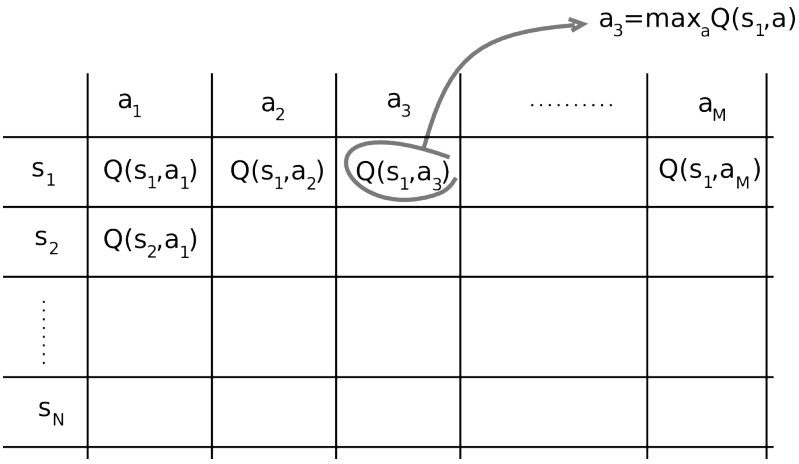


Figure 6.1: When reinforcement learning is applied in a setting with discrete actions and states a lookup-table like the one shown in this figure can be applied. This table stores the Q-values for every pair action-state. Given any state s, the greedy policy, i.e. the optimal policy according to what has been learnt so far, would involve selecting the action with the highest Q-value.

If the optimal action-values are known, $Q^*(s,a)$, determining the optimal policy is straightforward in a setting with discrete actions, as an exhaustive search is possible. If both states and actions are discrete, the action-values can be represented by tables and it would be sufficient to select the action with the highest Q value, henceforth, picking the appropriate action reduced to a look-up (Fig. 6.1). Therefore now the problem is how to get these action-values.

6.2.4 Monte Carlo methods in reinforcement learning

In general, there is a wide variety of methods that attempt to estimate $V^*(s)$, or $Q^*(s,a)$. These methods can be split mainly intro three classes: (i) dynamic programming-based optimal control approaches such as policy iteration or value iteration, (ii) roll-out-based Monte Carlo methods and (iii) temporal difference methods such as $TD(\lambda)$, Q-learning, SARSA, etc [100, 99]. Basically, the robot begins with an initial set of random values, $Q(s,a) \in [-1,-0.95]$, $\forall s$, and then it initiates a exploration of its environment executing the control policy π. As the robot moves performing the control policy π, it continually makes predictions about when it will receive negative reinforcements, in such a way that later comparisons of the predictions and the rewards the robot actually received will allow the updating of the utility values $Q(s,a)$. As the learning progresses, the initially random values $Q(s,a)$ should converge towards the optimal ones $Q^*(s,a)$.

We will use Monte Carlo methods to learn the utility function, as it is one of the simplest strategies. Monte Carlo methods are methods that use randomness, the basic idea is to explore randomly (i.e. to sample the environment), and to update values based on actual sample returns. Usually Monte Carlo methods are defined only for episodic tasks, i.e., we assume that the experience (robot-environment interaction) is divided into episodes, and all episodes eventually terminate no matter what actions are selected. It is only upon the completion of an episode that value estimates and policies are changed. Monte Carlo methods are thus incremental in an episode-by-episode sense, but not in a step-by-step sense. In our case, and for the experiments described later in this chapter, we will assume that the episode ends once the robot receives negative reinforcements.

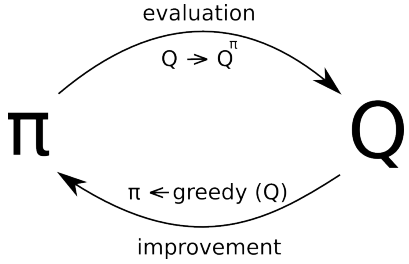


Figure 6.2: Schematic representation of the two stages in Monte Carlo methods. First the robot interacts with the environment collecting data (states, actions and rewards). This is the policy evaluation. After this data collection, the Q-values will be updated and hence the control policy can be improved. This is the policy improvement step. This process is repeated cyclically.

There are two types of Monte Carlo methods (MC) that can be applied in estimating $Q(s,a)$: The every-visit MC method and the first-visit MC method [100]. We have applied the every visit MC method. This method consists on two states (Fig. 6.2): the MC policy iteration, which is the the policy evaluation using the MC method, and the Policy improvement step, which builds the greedy policy considering the current estimate of the action-value function, according to algorithm 1.

Initialization:
begin
  Initialize, for all $s \in S$, $a \in A(s)$:
    $Q(s,a) \leftarrow arbitrary$;
    $\pi(s) \leftarrow arbitrary$;
    $Returns(s,a) \leftarrow emptylist$;
end
repeat
  Generate an episode using π
  forall *pairs* (s,a) *in the episode* **do**
    T_lapse $\leftarrow$ time interval since the occurrence of (s,a) to the negative reinforcements
    Append T_lapse to $Returns(s,a)$
    $Q(s,a) \leftarrow -e^{\frac{-average(Returns(s,a)}{50T}}$
  end
  forall *s in the episode* **do**
    $\pi(s) \leftarrow argmax_a Q(s,a)$
  end
until *end of the learning stage*;

Algorithm 1: Every-visit MC method.

There are other approaches, even for off-policy Monte Carlo Control or the ε-soft on-policy Monte Carlo control [100], nevertheless we decided to select the every visit strategy since it is one of the simplest alternatives, and we plan to combine learners to speed up the learning procedure.

6.2.5 Use of Fuzzy ART networks to represent the world

Our system needs to represent the world around the robot through a finite set of states. In our case, the system will use *Fuzzy ART neural networks* [104] to build a representation of the environment that will dynamically increase to include new situations that have not been seen before. This set of states must be precise enough to avoid perceptual aliasing (i.e. situations that require different actions but which are codified as the same state); on the other hand, the set of states should be as small as possible since the learning time increases exponentially with the number of states. Therefore, the design of the set of states is a delicate task.

If we really want a robot able to adapt to its workspace without human help, we must expect that the robot will encounter unforeseen situations quite regularly during the learning process. In consequence, we must avoid offline-created sensor-state mappings and, on the contrary, use strategies able to identify important events in the stream of sensor inputs in a largely unsupervised way. We need a robot able to build a sensor-state mapping that increases dynamically to include new situations that have not seen before. This is the reason why we suggest the use of Fuzzy ART artificial neural network to obtain this dynamic creation of the set of states as the robot explores the environment. In general we can assume that a FuzzyART network is able to cluster the robot sensor readings into a finite number of distinguishable situations that we call *states*. To do this, the FuzzyART divides the sensor space into a set of regions (Vector Quantization). Each one of these regions will have a prototype representing it. The FuzzyART works on the idea of comparing the input information with the prototypes of the regions into which the network has divided the sensor space so far. If there is a prototype that is similar enough to the input pattern, it is considered that there is *resonance*, and the network will consider that this input pattern belongs to the region represented by the prototype which has resounded. In this case the network will only perform a slight update of the prototype, so that it incorporates some characteristics of the input data. When the input pattern does not resound with any of the stored prototypes (states), the network creates a new region using the input pattern as its prototype.

The input of the Fuzzy ART will be an M-dimensional vector containing sensor readings. Each one of these components will be translated into the interval $[0,1]$. To prevent the Fuzzy ART from creating too many states, we will normalize the inputs using *complement coding*. The complement coding doubles the size of the input vector, i.e. the complemented coded input I to the recognition system happens to be a 2M-dimensional vector:

$$I = (a, a^c) \equiv (a_1, \ldots, a_M, a_1^c, \ldots, a_M^c),$$

where $a_n^c = 1 - a_n$. Using complement coding, the norm of the input vector will always equal the dimension of the original vector (M). The prototypes of the states (i.e. prototype that identifies each state learnt by the Fuzzy ART network), will be codified as an array of M dimensions with values in $[0,1]$: $W_j = (w_{j1}, w_{j2}, \ldots, w_{jM})$, where the sub-index j refers to the state. The behaviour of the Fuzz ART is determined by two parameters: learning rate $\beta \in [0,1]$; and a vigilance parameter $\rho \in [0,1]$. The way the Fuzzy ART network operates can be summarized in the following steps (there are some important differences in comparison with the general proposal described in [104]):

1. After presenting an input **I** to the network, there will be a competitive process after which the categories will be sorted from the lowest activation to the highest. For each input **I** and each state (category) j, the activation function T_j is defined as:

$$T_j(I) = \frac{|I \wedge w_j|}{|w_j|} \tag{6.14}$$

   the fuzzy operator AND $\wedge$ is $(x \wedge y)_i \equiv min(x_i, y_i)$ and the norm $|\bullet|$ is defined as:

$$|x| \equiv \sum_{i=1}^{M} |x_i| \tag{6.15}$$

2. the state with the maximum activation value will be selected to see if it resounds with the input pattern **I**

$$J = arg\_max_j\{T_j : j = 1, ..., N\} \tag{6.16}$$

3. The Fuzzy ART network will enter in resonance if the matching between the input **I** and the winning state **J** is greater or equal than the vigilance parameter ρ:

$$|I \wedge w_J| \geq \rho |I| \tag{6.17}$$

   If this relation is not satisfied, a new state will be created and the new prototype vector will be equal to the input **I**.

4. When the network enters in resonance with one input, the prototype vector w_J is updated

$$w_j^{(new)} = \beta I + (1 - \beta) w_j^{(old)}. \tag{6.18}$$

6.2.6 Teachable robots: providing reinforcement to learn

According to psychology theories, learning is strengthened if it is followed by positive reinforcement "pleasure" and weakened if it is followed by punishment "pain" [105]. This is something that is clearly described in Thorndike's Law of Effect [106]. Inspired by these psychological theories, our robot will learn from a feedback (reinforcement) that somehow evaluates its interaction with the environment. Nevertheless, getting this feedback is still an important drawback: we cannot ask people without knowledge of robots to provide a set of rules which specify when the robot is doing right or wrong.

A key aspect to achieve successful service robots lies in the possibility that any person (not necessarily specialist in robotics) can teach robots new tasks, or can do something that will alter the behaviour of the robot. An article published in Artificial Intelligence regarding *how the humans want to teach the machines* [107], comes to the conclusion that the processes of *teaching and learning* must be closely linked. A good *instructor* must hold a mental model of the state of the learner (trying to guess what the robot has understood, what the robot knows and what the robot is still ignorant of, etc). On the other hand the robot must help the instructor doing the learning process as clear as possible. In short, the process of teaching/learning must be bi-directional.

Taking this into account we decided that the reinforcement should come from a human observer that is seeing what the robot does. This observer will be able to punish the robot by simply pressing a button in a wireless joystick. This action will be enough to tell the robot that what it is doing is unsatisfactory. It is important to be aware of the fact that this way of providing the reinforcement is highly not deterministic, i.e., the same user can give the robot negative reinforcements in certain situations but remain impassive in other scenarios that are very similar. Moreover, the human observer can change his mind about what is right or wrong while the robot is still learning. When the user presses the button of the wireless joystick to give the robot negative reinforcement, the robot learns from it and transfers the control to the joystick of the human user, so that the user will be able to move the robot and place it in a suitable position to go on learning, once this manual control is over, the user will press a second button to continue the learning process.

6.2.7 Pyramid representation of the set of actions

According to the explanation provided in the previous sections, our robot will learn an utility function of states and actions $Q(s,a)$ using the Monte Carlo algorithm. Nevertheless this involves the existence discrete sets of states and actions (lookup table shown in Fig. 6.1). Regarding the discrete representation of actions, there are some issues that must taken into account: if the set of actions is obtained as a thin partition of the action space, the robot could determine the best control policy with a high accuracy. Nevertheless, due to the fact that the number of actions is high, the learning process could take too long and, in the worst case, might not converge at all. On the other end, if the set of actions is obtained as a rough partition of the action space, the learning process will take shorter, but there could be convergence problems due to the fact that the partition of actions might be too coarse and thus the robot cannot attain the right and specific action for some of the states. Due to this, we opted for an intermediate alternative that will allow a gradual and incremental learning of which is the best action for each state, in particular we will inspire in what is called *Spatial Pyramid Representation* [108, 109] in computer vision.

The Spatial Pyramid Representation [108, 109] in computer vision is a widely used method for embedding both global and local spatial information of an image into a feature vector. In particular the image is divided into a sequence of increasingly finer grids on each pyramid level. The features are extracted from all the grid cells and are concatenated to form one huge feature vector.

We inspired in this kind of strategy to divide the action space into a set of increasingly finer intervals. The spatial pyramid representation in computer vision divides an image (which is a two dimensional entity) into a set of regions, but this strategy is not valid for a multidimensional action space (where the number of dimension can be higher than two). To solve this we have developed our own strategy to achieve a pyramidal representation of the set of actions: in particular we used a connected tree where each level of the tree divides the action space into a set of Voronoi regions (Fig. 6.3). Basically, as we can see in this figure, every level of the tree divides the action space into a set of regions that are smaller as we move further in the tree, i.e. as we move down in the tree the number of Voronoi regions is higher and their size is smaller. Finally, this tree is binary, i.e, each node has only two successors. Considering that a^i_j represents the centroid of the j^{th} Voronoi region in which level i partitions the action space, it is true to say that the successors of this node verify that their Voronoi centres are within

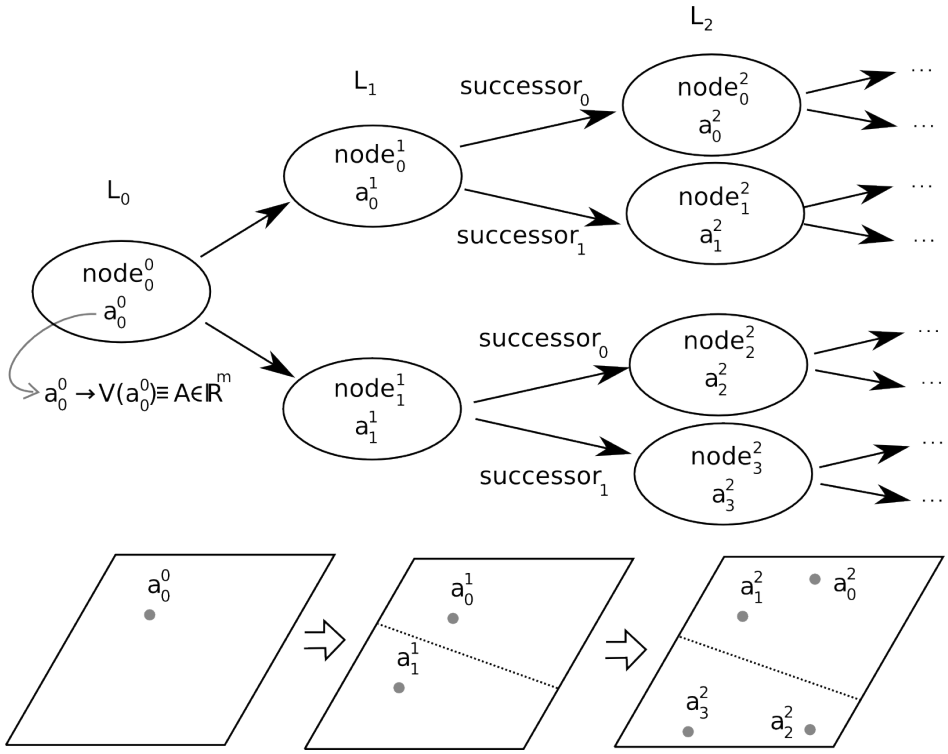


Figure 6.3: Pyramid representation of the action space. This representation is obtained at the beginning of the learning process. Each node of the tree represents a Voronoi region, the centre of which is represented in the node (a^i_j). Each level of the tree divides the action space in regions that are smaller as we move further in the tree.

the Voronoi region of the parent node. Thus, in the example shown in Fig. 6.3, $a^2_0 \in Vor(a^1_0)$, $a^2_1 \in Vor(a^1_0)$, $a^2_2 \in Vor(a^1_1)$, $a^2_3 \in Vor(a^1_1)$, and so on.

Initialization
$a_0^0 = random\_action \in A \subset R^m$, being m the dimension of the action space
$num\_succ(a_0^0) = 0$
for *level=1* **to** L **do**
  $nodes\_in\_level(l) = 0$
  while $nodes\_in\_level(l) < 2^l$ **do**
    $a = random\_action \in A$
    $BMU = arg\_min_j d(a_j^{l-1}, a)$
    if $num\_succ(BMU) < 2$ **then**
      accept a as one of the Voronoi centres in level l
      $a^l_{2*BMU+num\_succ(BMU)} = a$
      $num\_succ(BMU) = num\_succ(BMU) + 1$
      $nodes\_in\_level(l) = nodes\_in\_level(l) + 1$
    end
  end
end

Algorithm 2: Random partitioning of the action space.

Now, instead of having a Q value for every possible action, our algorithm will keep a Q value for every Voronoi region, these Q values will evolve during the learning procedure. The value $Q(s, Vor(a_j^i))$ represents the *expected time interval* before a robot failure when the robot performs any action included in $Vor(a_j^i)$ in state s.

The reason why we will use the name "pyramid representation" of the set of actions, is because the number of successors for every node could be higher than two and, on the other side, similar to what happens with the feature vector in the case of the images, the Q-value from all the nodes are concatenated to form one *utility vector* which will be used to determine the action the robot will perform at every state.

This pyramid representation is obtained only once, at the beginning of the learning process, before the robot starts interacting with the environment.

6.2.8 Use of the pyramid representation to determine the action to be taken

This pyramid representation (binary tree) can be used to decide the action the robot should carry out at every instant. Nevertheless, in this case the decision process will be a sequential procedure (an L-step procedure, being L the number of levels of the pyramid).

$best\_node = node_0^0$
$best\_set\_of\_actions = Vor(a_0^0)$
for *level=1* **to** L **do**
  the successor with the best Q value is chosen
  $best\_successor = arg\_max_{i=0,1} Q(s, Vor(successor_i(best\_node)))$;
  the best set of actions is updated according to the chosen successor
  $best\_set\_of\_actions = best\_set\_of\_actions \bigcap Vor(best\_sucessor)$;
  $best\_node = best\_successor$
end

Algorithm 3: Sequential process to determine the action to be taken.

According to algorithm 3 $best\_set\_of\_actions$, is the Voronoi region that contains all the possible actions that can be performed by the robot. Nevertheless, if we consider that the intersection of Voronoi regions is a Voronoi region, is straightforward to deduced that the Voronoi region represented by $best\_set\_of\_actions$ is reduced progressively during the sequential procedure described in this algorithm.

Because the decision on the action to be taken is a sequential procedure, if we select a node (parent node in a particular level), when the successor is chosen, the set of best actions is the intersection of the Voronoi region of the parent node and the Voronoi region corresponding to the successor, therefore it will mean that the action the robot finally performs is within the Voronoi region of the parent node and the successor. Thus, due to this sequential procedure it can be assumed that each level of the pyramid representation subdivides the Voronoi partition of the parent level into a random set of sub-partitions, so that each Voronoi region of the parent level will be split into two other Voronoi regions.

6.2.9 Use of the pyramid representation to determine the greedy policy

As the learning progresses the robot should tend to take more and more actions that are greedy, i.e. which maximise the time interval before a robot failure. Let us remember that $Q(s,Vor(a_j^i))$ represents the *expected time interval* before a robot failure when the robot performs any action included in $Vor(a_j^i)$ in state s. Nevertheless neither these values, nor the action-decision procedure described before, let us guarantee a good convergence of the learning procedure. To understand this it is enough to consider that when there is a Voronoi region which is too big, and therefore contains too many actions that might be inappropriate for a particular state, its corresponding Q value will tend to be low, no matter that there might be a very good action inside this region. This could significantly slow the learning process or even cause a convergence towards suboptimal actions. To prevent this problem, we will use another value, which we will call *best time before a robot failure*. By the $best\_Tbf(Vor(a_j^i))$ we represent the highest time before a robot failure which can be obtained with an action included in $Vor(a_j^i)$, i.e., if we compute the *expected time interval* before a robot failure for every action included in $Vor(a_j^i)$, then $best\_Tbf(Vor(a_j^i))$ would be the highest of these times.

As in the case of the expected times before a robot failure (Tbf), it is possible to project the $best\_Tbf()$ in the interval $[-1,0]$:

$$best\_Q(Vor(a_j^i)) = -e^{(-best\_Tbf(Vor(a_j^i))/50T)}$$

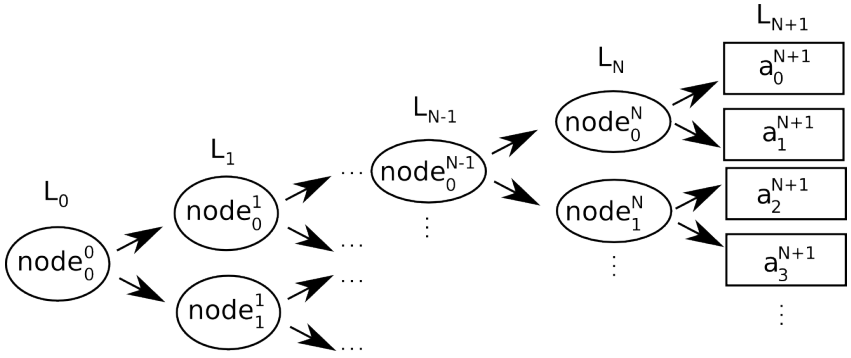


Figure 6.4: Pyramid representation of the action space. The information (Q-values) associated to the nodes in this representation will be used to determine two new values for every node: best_Q and best_Tbf.

Let us consider the example shown in Fig. 6.5 to understand how these *best times before failure* and their corresponding Q-values are computed. This figure shows a pyramid representation for an action space A. According to this figure, $node_0$ in L_N has two successors (i.e. actions associated to it): a_0^{N+1} and a_1^{N+1}. Since this is the last level of the pyramid, $Vor(a_0^{N+1})$ and $Vor(a_1^{N+1})$ are the smallest Voronoi regions, i.e, given any state s we only know the consequences of executing actions that are either a_0^{N+1} or very similar, and the consequences of executing a_1^{N+1} or similar actions (those included in $Vor(a_1^{N+1})$). These consequences are reflected in $Q(s,Vor(a_0^{N+1})$ and $Q(s,Vor(a_1^{N+1})$. Therefore, these Q-values represent the most detailed information we have, and hence we can assume that the $best\_Q$ for $node_0$ in L_N can be approximated as:

$$best\_Q(s,node_0^N = Vor(a_0^N)) = max\{Q(s,Vor(a_0^{N+1})),Q(s,Vor(a_1^{N+1}))\}$$

and therefore:

$$best\_Tbf(s,node_0^N = Vor(a_0^N)) = \ldots$$
$$-50.T.ln(-max\{Q(s,Vor(a_0^{N+1})),Q(s,Vor(a_1^{N+1}))\}$$

Similarly, if for any state s we have computed the $best\_Q$ for all nodes in the N^th level of the pyramid, we can estimate the $best\_Q$ for the nodes in the $N-1$ level:

$$best\_Q(s,node_0^{N-1}) = \ldots$$
$$max\{best\_Q(s,succ_0(node_0^{N-1})),best\_Q(s,succ_1(node_0^{N-1}))\}$$
$$best\_Q(s,node_1^{N-1}) = \ldots$$
$$max\{best\_Q(s,succ_0(node_1^{N-1})),best\_Q(s,succ_1(node_1^{N-1}))\}$$
$$\ldots$$

where $successor_0$ of $node_0(L_{N-1})$ is $node_0(L_N)$ and $successor_1$ of node $node_0(L_{N-1})$ is $node_1(L_N)$), and so on. This equation is true for the remaining nodes in the three, which would allow a recursive computation of all of them.

It is important to be aware of the fact that $best\_Q(s,...)$ for every node in the tree would be one of the Q-values stored at the last level of the pyramid. The problem is that the update of the Q values at the last level will be very slow, which means that most of the $best\_Q()$ will be very noisy and erratic. To solve this issue and accelerate the learning process we use algorithm 4.

$best\_Tbf$ is computed for the nodes in the last level of the tree,
for *for i=0* **to** $num\_nodes\_level(L)$ **do**
  $best\_Tbf(s, node_i^L) = -50.T.ln(-max\{Q(s, Vor(succ_0)), Q(s, Vor(succ_1))\})$
end
$best\_Tbf$ is computed for the rest of nodes in the tree,
The computation is done backwards, from the penultimate level to the first one
for *for l=L-1* **to** *0* **do**
  for *for i=0* **to** $num\_nodes\_level(l)$ **do**
    the $best\_Tbf$ for the branch 0 is computed
    $best\_Tbf_0 = \frac{-50.T.ln(-Q(s,Vor(succ_0)))+(L-l)*best\_Tbf(s,succ_0)}{1+(L-l)}$;
    the $best\_Tbf$ for the branch 1 is computed
    $best\_Tbf_1 = \frac{-50.T.ln(Q(s,Vor(succ_1)))+(L-l)*best\_Tbf(s,succ_1)}{1+(L-l)}$;
    the $best\_Tbf$ for the node being analysed is estimated
    $best\_Tbf(s, node_i^l) = \ldots$
    $max\{best\_Tbf_0, best\_Tbf_1\}$
  end
end

Algorithm 4: Estimation of the best_Tbf values for the pyramid representation.

In this algorithm we can realize that all Q values are used to compute the $best\_Tbf$. Nevertheless, instead of working in the time domain, which might involve working with very big numbers when the learning converges to a good control policy, we prefer to work only with the Q-values, therefore, the previous algorithm is translated into the following procedure:

best_Q is computed for the nodes in the last level of the tree,
for *for i=0* **to** $num\_nodes\_level(L)$ **do**
  $best\_Q(s, node_i^L) = max\{Q(s, Vor(succ_0)), Q(s, Vor(succ_1))\}$
end
best_Q is computed for the rest of nodes in the tree,
The computation is done backwards, from the penultimate level to the first one
for *for l=L-1* **to** *0* **do**
  for *for i=0* **to** $num\_nodes\_level(l)$ **do**
    the *best_Q* for the branch 0 is computed
    $best\_Q_0 = -(-Q(s, Vor(succ_0)).(-best\_Q(s, succ_0))^{(L-l)})^{1+(L-l)}$;
    the *best_Q* for the branch 1 is computed
    $best\_Q_1 = -(-Q(s, Vor(succ_1)).(-best\_Q(s, succ_1))^{(L-l)})^{1+(L-l)}$;
    the *best_Q* for the node being analysed is estimated
    $best\_Q(s, node_i^l) = \ldots$
    $max\{best\_Q_0, best\_Q_1\}$
  end
end

Obviously, it is possible to incorporate these new values $best\_Q()$ during the sequential action decision process, and thus avoid the potential convergence problems mentioned at the beginning of this section.

$best\_node = node_0^0$
$best\_set\_of\_actions = Vor(a_0^0)$
for *level=1* **to** L **do**
  the successor with the *best_Q* is chosen
  $best\_sucessor = arg\_max_{i=0,1}\{best\_Q(s, sucessor_i(best\_node))\}$;
  the best set of actions is updated according to the chosen successor
  $best\_set\_of\_actions = best\_set\_of\_actions \bigcap Vor(best\_sucessor)$;
  $best\_node = best\_successor$
end

6.2.10 Use of ensembles to learn

Speeding up the learning procedure usually involves using less and less data. Therefore there is an increasing risk of over fitting the available set of examples (low bias), but compromising the generalization (high variance), i.e, the robot shows inappropriate or unexpected behaviours when there are slight changes in the environment.

As it was described before, the robot learns a function $Q(S \times A)$ based on a data set collected while the robot is moving $(s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+a}, \ldots)$. This Q function predicts when the robot will make a mistake, i.e., when the robot will receive negative reinforcements after the execution of each action. Following [110], we will indicate the dependence of the predictor Q on the training data by writing $Q(S \times A; D)$, instead of $Q(S \times A)$. The mean squared error of Q as predictor of the real time before a robot failure (Tbf) may be written as:

$$E_D[(Q(S \times A; D) - Q^*[S \times A])^2]$$

where E_D is the expectation operator with respect to the data set D, and $Q^*[S \times A]$ is the target function, i.e., the real time before failure for every pair $s \in S$ and $a \in A$ when the greedy policy is being used to control the robot. We can break the squared errors in two components (bias and variance):

$$\begin{aligned} E_D[(Q(S \times A; D) - Q^*[S \times A])^2] = \ldots \\ (E_D[Q(S \times A; D)] - Q^*[S \times A])^2 \text{“}bias'' \\ + E_D[(Q(S \times A; D) - E_D[Q(S \times A; D)])^2] \text{“}variance'' \end{aligned}$$

The *bias* represents how much the predictor Q fits the data set D, i.e., a low bias means that the predictor resembles very accurately the data set used to get it (D). The variance is a measure of the extent to which the prediction is sensitive to the data it was obtained from, i.e., the extent to which the same results would have been obtained if a different set of data was used. Therefore a low variance implies a good generalization.

Usually there is a trade-off between bias and variance;attempts to decrease the bias are likely to result in high variances, while efforts to decrease the variance usually result in increased bias. A strategy for reducing both, the bias and the variance, is to use an ensemble of predictors [111] $(Q1, Q2, ..., QN)$.

As we can see in Fig. 6.5, to get a low bias low variant robot controller we decided to build an ensemble of independent learners (i.e. predictors of the time before a robot failure). For generalization purposes, each predictor will have its own state representation (fuzzy ART)

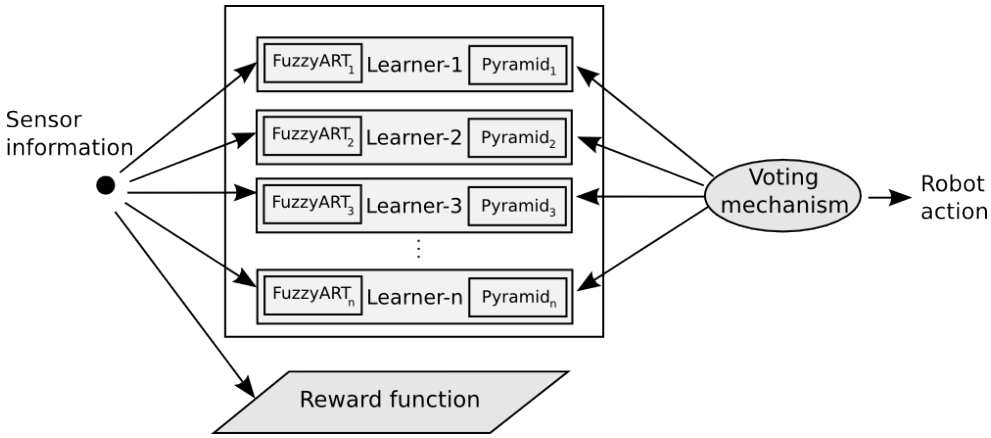


Figure 6.5: Ensemble of independent learners to achieve fast learning processes. Every learner uses its own representation of the environment (Fuzzy ART), and its own pyramid representation of the action space.

and its own action representation. The major difficulty when combining expert opinions is that these opinions tend to be correlated or dependent. This is the reason why, to avoid or reduce this artefact, we had decided to use different ART networks (with different and random vigilance parameters), and different pyramid representations for every learner. Each predictor will have to learn the Q-values for every state, as it was explained in the previous sections. Nevertheless, the best action to be executed at every state will be inferred from the joint participation of the the predictors. In particular, given a set of candidate actions that can be performed by the robot, each predictor will issue a vote for each one of these actions. The best-valued action will be the one that the robot finally performs. This process will be sequential in the sense that, initially, every learner will issue the votes considering the information (Q-values) associated to the first level of the pyramid representation. After the voting procedure only those candidate actions that received the highest votes will pass to a second stage where, once again, every learner will issue a new vote. In this second stage, every learner will use the information (Q-values) associated to the second level of the pyramid to vote for every candidate action that passed the filter of the first level. Once again, the set of candidate actions will be reduced to those which receive the highest votes. The process will be repeated consecutively using, in every step, the information associated to a different layer of the pyramid representations.

Let us consider a set of candidate actions to be performed by the robot. Initially any of these actions can be selected: $candidate[j] = 1, \forall j$ being j the index of the action

for *i=1* **to** *number_of_learners* **do**

BMU represents the node of the i learner that will vote for action j Since in the first layer there is only one node ($node_0^{l=0}$), $BMU(i, j) = 0$;

end

for *l=1* **to** L **do**

$w[j] = 0, \forall j$

for *i=1* **to** *number_of_learners* **do**

for *j=1* **to** *number_of_actions_being_analysed* **do**

if $candidate[j] == 1$ **then**

next_node represents the successor of $BMU(i)$ whose Voronoi region contains the action being analysed

$w[j] = w[j] + \frac{-50.T.ln(-Q(s,next\_node))+best\_Tbf(s,next\_node)}{2}$

we update the node that will issue the vote in the next level for learner i and action j. $BMU(i, j) = next\_node$

end

end

end

Actions which did not receive the highest weight are removed

$wmax = arg\_max_{j=1,...,number\_of\_actions\_being\_analysed}\{w[j]\}$ **for** *j=1* **to** *number_of_actions_being_analysed* **do**

if $w[j] = wmax$ **then**

$candidate[j] = 1$

else

$candidate[j] = 0$

end

end

end

The action to be executed is randomly chosen amongst the final subset of candidates.

Algorithm 5: Decision process using an ensemble of learners. Best_Tbf values are used.

Similarly to what we did in the first section, for clarity purposes we showed the voting procedure considering the expected times before a robot failure, nevertheless, it is better to work with the Q-values, since this is the information being stored by our algorithm, and it avoids the use of big numbers when the learning process is converging to good control policies, or during continuous learning procedures:

Let us consider a set of candidate actions to be performed by the robot. Initially any of these actions can be selected: $candidate[j] = 1, \forall j$, being j the index of the action

for *i=1* **to** *number_of_learners* **do**

BMU represents the node of the i learner that will vote for action j. Since in the first layer there is only one node ($node_0^{l=0}$), $BMU(i,j) = 0$;

end

for *l=1* **to** L **do**

$w[j] = 1, \forall j$

for *i=1* **to** *number_of_learners* **do**

for *j=1* **to** *number_of_actions_being_analysed* **do**

if $candidate[j] == 1$ **then**

next_node represents the successor of $BMU(i)$ whose Voronoi region contains the action being analysed

$w[j] = w[j] * -1 * \{-Q(s,next\_node) * -best\_Q(s,next\_node)\}^{1/2}$

we update the node that will issue the vote in the next level for learner i and action j. $BMU(i,j) = next\_node$

end

end

end

Actions which did not receive the highest weight are removed

$wmax = arg\_max_{j=1,\ldots,number\_of\_actions\_being\_analysed}\{w[j]\}$ **for** *j=1* **to** *number_of_actions_being_analysed* **do**

if $w[j] = wmax$ **then**

$candidate[j] = 1$

else

$candidate[j] = 0$

end

end

end

The action to be executed is randomly chosen amongst the final subset of candidates.

Algorithm 6: Decision process using an ensemble of learners. Best_Q values are used.

6.3 Experimental results

In this section we describe several experiments that we have carried out to test the validity of our learning algorithm, which take place at the 100sqm robotics laboratory of our research centre (CITIUS). We have used the same parameters for our algorithm in all the experiments that we describe in this section:

- The number of learners in the ensemble is 100. We tested with different values but the results are very similar.
- The number of levels in the pyramid is 3. We have also made some tests with values up to 6 but it the results are similar.
- Vigilance parameter (ρ) is a random number between 0.92 and 0.97.
- The robot's angular velocity must be between -0.7859 and 0.7859 radians per second.

The experiments are divided in two parts: first, we started testing our proposal to learn a wall following controller, and then, we moved forward to our original goal, the learning of a human following controller.

The learning of a wall following controller is a familiar task in our research group, therefore we found it interesting to start testing our proposal with this problem to validate our proposal. In this task the robot starts inside a room with different types of corners and needs to learn how to follow the wall in order to autonomously complete several laps to the room without colliding with the walls but as close as possible to them. The inputs of our algorithm for this task are the 270 laser measurements of our robot: the resolution is 1 degree and the field of view goes from -45 to 225 degrees. We have performed two kinds of wall following experiments:

a) Learning the most appropriate angular velocities for each situation while using a fixed linear velocity of 0.15 meters/sec. For this set of experiments we have set up a room with several narrow corners of different types as it can be seen in the picture of Fig. 6.6.

   The experiments that we have carried out validated the learning abilities of our robot: it was able to learn the task in the first lap, while it only needed very few reinforcements in the second and third laps to get ready to autonomously perform several laps, which is the point where we can conclude that it has learnt the wall following task with fixed

Figure 6.6: Learning environment for the wall following task with fixed linear velocity.

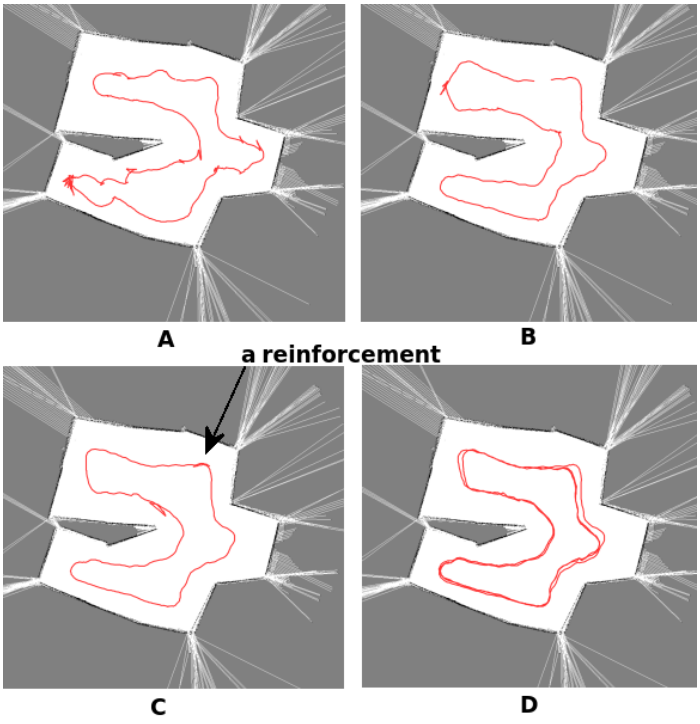


Figure 6.7: Trajectories performed by our robot while learning the wall following task with fixed linear velocity. In laps A to C the robot learns the task while in figure D the robot autonomously performs three laps.

linear velocity. In this regard, Fig. 6.7 shows the trajectories of our robot during one of the experiments. In these trajectories we can even observe the points where it received a reinforcement.

Figure 6.8: Wall following environment for the second batch of tests of our learning algorithm.

b) Learning both angular and linear velocities of our robot. In this set of experiments we removed the fixed linear velocity to check if our robot was still able to learn the same task when it had to learn to variables at the same time. For this purpose we changed the action space which was composed by angular velocities to pairs of linear and angular velocity, but we had to include the following restrictions due to the robot's characteristics: linear velocity between 0.0 and 0.6 m/s and linear velocity $\leq (-0.5|angular\_vel|)/0.78 + 0.6$. Moreover, we changed the wall following circuit a bit to create bigger spaces in which the robot should choose high speed and others where it should choose slow speed (Fig. 6.8).

Once again, the experiments that we have carried out confirmed the validity of our algorithm. It was able to learn the task in the first two or three laps, and then it needed from 2 to 6 more laps with a few reinforcements to autonomously complete three consecutive laps without any reinforcement. In this regard Fig. 6.9 shows the speed improvement that was achieved with our algorithm when compared with the speed in the previous experi-

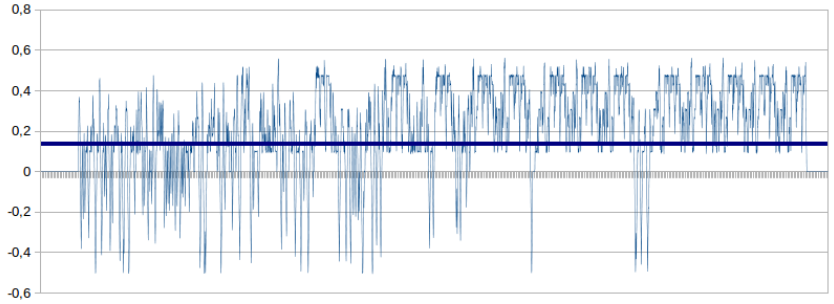


Figure 6.9: Linear velocity that was achieved by our robot while learning the wall following behaviour. The blue line corresponds to the previous fixed linear velocity. The negative values correspond to the moments in which the robot received a reinforcement and had to be repositioned.

ments. Moreover, in Fig. 6.10 we can see the trajectories of our robot during one of these experiments.

These experiments proved us that our algorithm was ready for learning more complex tasks like the following of a human, which requires the learning of linear and angular velocities at the same time. Therefore we cleaned up our laboratory to test the learning of the human follower controller in a environment without obstacles (Fig. 6.11). In this first batch of experiments the inputs of our algorithm were the distance of the robot's target and his angle with respect to the robot's front direction.

In these experiments we observed how the robot was able to learn the human following ability quickly, while it could take a bit more to learn the *stop in front a standing human* ability. Fig. 6.12 shows several statistics collected during one of the experiments where we first learnt how to follow a human and then the stop behaviour for the case that the human does not move. Nevertheless the experiments that we have carried out proved that the algorithm that we have designed is valid for the learning of this task.

Finally, we wanted to test our algorithm in a more real situation: human following while avoiding obstacles. For this reason, we set up a new environment in which we put a few obstacles (Fig 6.13). This time the human will walk around the obstacles while the robot learns to follow him. In this new set of experiments we seek that the robot learned how to avoid obstacles even if that meant to lose sight of the human for a few seconds, that is, the robot should learn how to recover the sight of the human once he has avoided the obstacle. The inputs were the same than in the previous set of experiments, but we included information

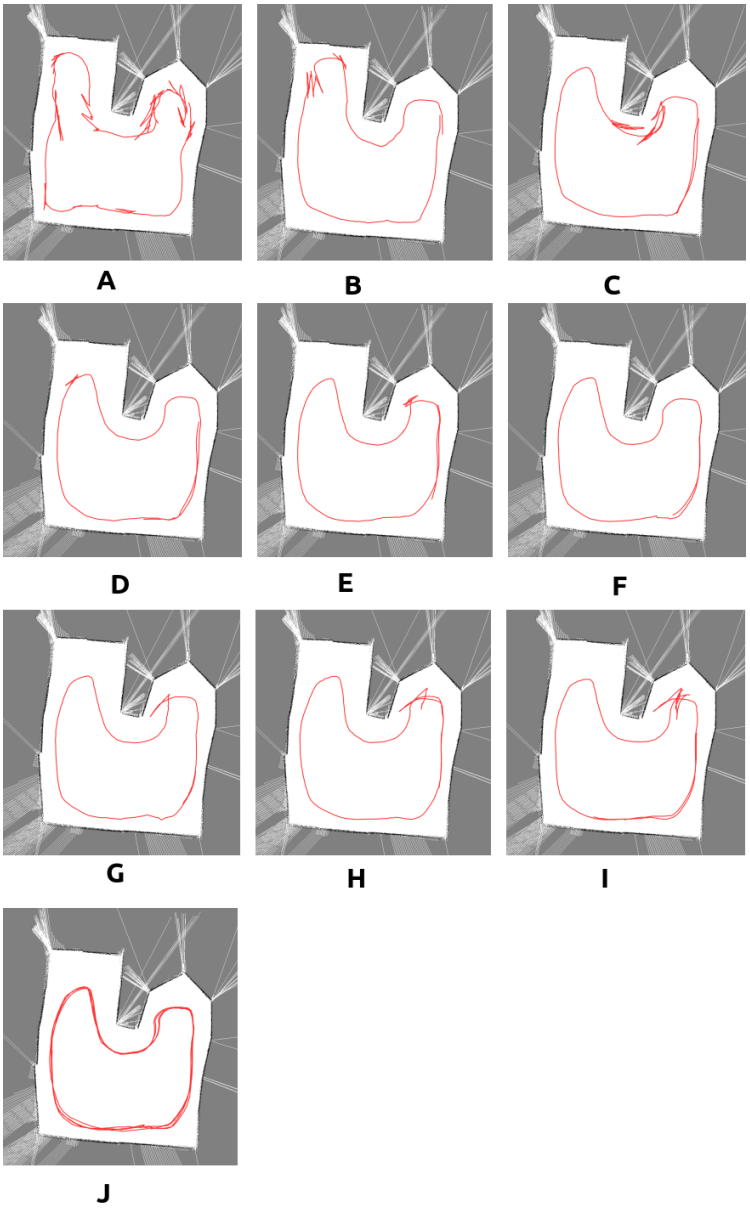


Figure 6.10: Trajectories followed by our robot to learn the wall following task. Both linear and angular velocities are learnt by the algorithm. Lap A,B and C concentrates most reinforcements, while laps D to I receive few of them. Figure J shows the final three autonomous laps performed by the robot.

Figure 6.11: Learning the human follower controller without obstacles in our laboratory.

about the obstacles: we calculated a repulsive force using the data from the laser scanner (in the same way than in the ad-hoc controller).

In these experiments we learnt that the best strategy was to start teaching the robot how to follow a human without obstacles, which is achieved in few minutes. Then, the human could start walking around the obstacles to allow the robot to learn how to avoid them. We need a few more minutes until the robot was able to learn the full task.

In the end, we can conclude that our robot was able to learn how to follow a human while avoiding obstacles that caused the robot to lose sight of the person but then the robot could recover from that situation and keep following the person. In this regard Fig. 6.14 shows the robot's trajectory for this experiment.

6.4 Conclusions

In the future robot systems will perform increasingly complex tasks in decreasingly well-structured and known environments. Robots will need to adapt their hardware and software, first only to foreseen, but ultimately to more complex changes of the environment. Most service robots should be constant learners: it should acquire new skills in an active, open-

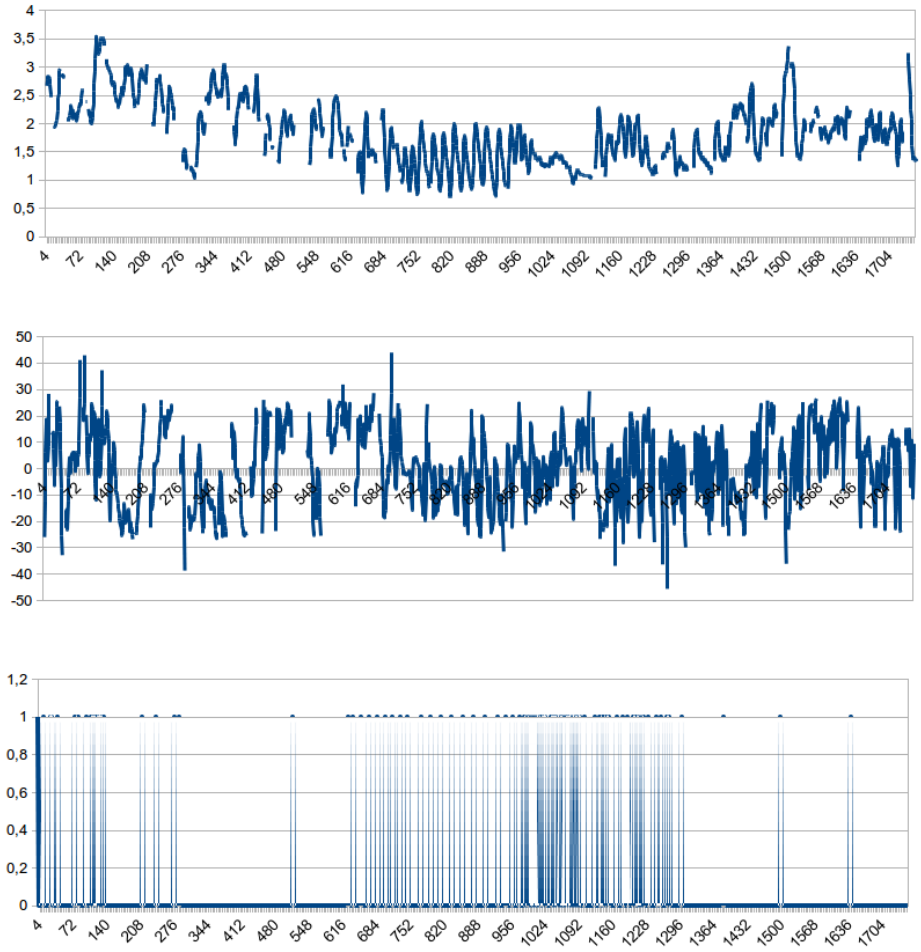


Figure 6.12: Statistics collected during the learning of the human follower controller without obstacles, in which the robot was learning how to follow a human (from the y-values 0 to 276 approx) and then how to stop in front of a human who is not moving (from the y-values 616 to 1296 approx). The top graph shows the distance in meters of the human from the robot. The graph in the middle shows the angle of the human given that zero corresponds to the front direction of the robot. The bottom graph shows the instants where the robot received a reinforcement.

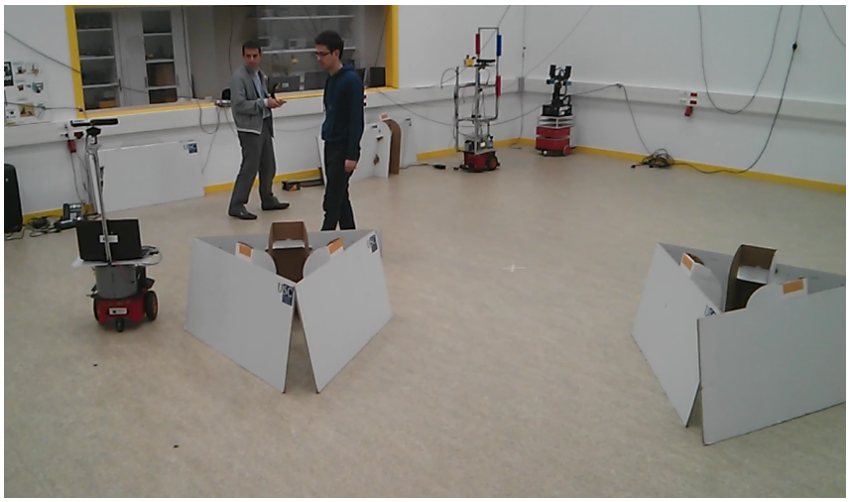


Figure 6.13: Learning the human follower with obstacles in our laboratory.

ended way, and develop as a result of constant interaction and co-operation with humans. Future service robots will be required to run autonomously over really long periods of time in environments that change over time. Examples include security robots, robotic caregivers, tour guides, etc. These robots will be required to live together with people, and to adapt to the changes that people make to the world. Hence, if a robot needs to adapt to such changes then life-long learning is essential.

In this chapter we have described a learning strategy based on reinforcement which allows fast robot learning from scratch using only its interaction with the environment. The strategy we have developed has the following characteristics:

1. Use of an ensemble of learners able to forecast the time interval before a robot failure. The use of an ensemble helps to get better generalization. The combination of all the members overcomes individual errors (the decision of the action to be performed by the robot at every instant is decided after a voting procedure). To facilitate this generalization, each member of the committee uses its own representation of the action space and the environment (set of states). Finally, the use of a committee also facilitates the action decision procedure and lowers the necessity of an explicit exploration strategy.

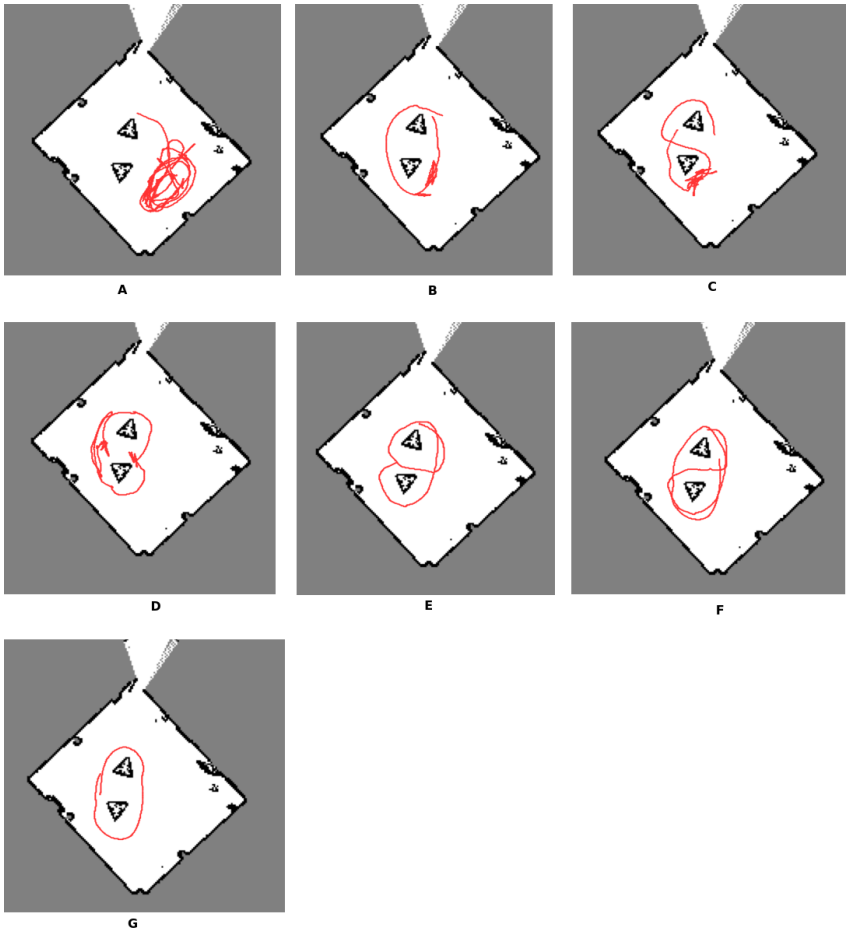


Figure 6.14: Trajectories performed by our robot while learning the task of following a human while avoiding obstacles. Figure A shows the part where the robot received many reinforcements to teach him how to follow a human without obstacles. Figures B to D show several laps around the obstacles where the robot learnt how to avoid them and keep following the human. Finally, figures E to G show how the robot followed a human among the obstacles with very few or no reinforcements at all.

2. Use of pyramid representations of the action space (binary tree in this case). We call it pyramid because in the general case it would not have to be a binary tree (i.e. the number of successors could be higher than two). On the other hand, and like in computer vision, the joint consideration of the Q-values associated to all levels of the tree influence the decision of the action to be taken by the robot. The use of this pyramid representation allows a progressive learning of the task, i.e., the robot will discover first the big interval which contains the right actions for every state, but this knowledge will be progressively refined so that the robot finally knows the correct action or narrow interval of actions that are suitable for every state.

We have tested our strategy in real world experiments. In particular the robot had to learn the wall following behavior and the person following behaviour. In the case of the wall following behaviour, the robot had to learn the angular velocities in one set of experiments, and both the angular and linear velocities in a second set of experiments. Regarding the person following behaviour the robot learnt the task in an empty environment, but we also saw how our proposal was able to learn the same task when there were obstacles around the robot. It is important to realize that in all these experiments a human user provided the reinforcement with a joystick. This is another challenge since sometimes the user can change his mind (regarding when something is right or wrong) during the learning procedure, etc. Our proposal was robust enough to cope with this.

Chapter 7

Conclusions

The main goal of this thesis is to create a general purpose tour-guide robot. It should be easy to use and easy to deploy. Therefore, we have proposed that the robot should learn routes while following a human who acts as an instructor. That is, anyone should be able to bring our robot to an event, teach routes to it, and then showcase these routes to the visitors of the event. In this sense, the instructor, and the rest of the users of our robot, could be anyone without knowledge of robotics. Moreover, out robot should be able to cope with unexpected situations or challenges that might be present on the different environments where it can operate. For this reason, we have created three different stages or states in our robot:

1. First, the deployment stage. It can take up to a few hours, depending on the size of the environment where the robot operates. In this stage a human must teleoperate the robot in order to acquire an occupancy map of the environment as well as to collect wifi data for the positioning system.

2. Then, the learning stage. Typically, a member of the staff of the even where the robot operates should teach the robot the routes that will be later on showcased.

3. Finally, the showcasing stage. Even tough anyone could still teach new routes to the robot, in this stage, it is intended that the main task of our robot is to show the routes that it has already learnt.

These stages led us to identify to different roles for our robot:

- The instructor, who should have some basic knowledge of the robot's capabilities. Its main task is to teach new routes to the robot by asking it to follow them.
- The visitors of an event, who can have very limited knowledge of the robot's capabilities. They should only pay attention to what the robot offers them, and to select their the route of interest.

In order to achieve our goals, we have pursued the development of robot behaviours which are as independent as possible from the environment, and from the humans in those environments. That is, we have searched for adaptable algorithms, self-learning strategies, and multi-sensor approaches to critical tasks, etc. More specifically, the milestones that we have achieved with the development of our tour-guide robot are (in order of appearance):

A) An efficient and robust person-following behaviour for mobile robots. This behaviour includes:
   - An efficient and robust human detector for depth images. It is able to continuously detect stand-up humans regardless of their pose. This detector suits perfectly the constrains of a mobile robot: it does not rely on background subtraction, and it has a low computational burden.
   - An study on the most suitable features for human characterisation in mobile robots. First, we have reviewed a wide range of the most popular visual features with a low computational cost. Then, we have selected an small subset of those features, using the criteria of most information with least redundancy.
   - An strategy to adapt the recognition of the humans to each particular situation. We have presented an algorithm that is able to dynamically assign different weights to the features that describe the appearance of the humans. This weight will be determined by the discrimination ability of each feature. In this way, we are able to focus on those features which will help us the most on deciding whether a detected human is the robot's target or not.
   - An study on which is the best method to determine the discrimination ability of a feature. Following the previous point, we have also studied which method is the best option to determine how good a feature is to discriminate between the robot's target and the rest of the people.

B) A novel human-robot interaction scheme. This includes:

   – Algorithms for fast hand detection and tracking. We have presented an efficient strategy that is able to continuously search for the people's hands and track them when they are found. In order to do so, we rely exclusively on depth information.

   – An augmented reality graphical user interface. We have proposed a new concept of human-robot graphical interface. In this interface the robot's users can see what the robot sees, and continuously receive feedback of their actions, as well as, get information of what the robot expects from them.

   – An interaction process. We have adhered to the ISO IEC-9126-4 Standard, and hence during several test rounds, we have collected user's feedback to build an easy human-robot interaction.

C) A multi-sensorial algorithm for mobile robot localization. We have proposed an algorithm that, using a particle filter, is able to fuse information from many sensors. Some highlights of the algorithm are:

   – Fuses information from different sources with non-correlated errors, increasing the overall performance of the algorithm, specially in crowded environments where laser scanners offer poor performance.

   – It avoids the confusion between similar spaces, at the same time that it provides accurate localization estimates.

   – Scalability. The solution can handle a variable number of sensors, even if they have different data rates.

D) A route recording process. We have proposed a novel way of introducing routes in a tour-guide robot: an instructor will teach them to our robot, which will follow the instructor recording its path, as well as, other information that the instructor provides to the robot. The main advantages are:

   – Allows anyone to introduce new routes to the robot, or to modify existing ones.

   – Reduces the expert's intervention to a minimum when deploying a new tour-guide robot.

E) A route reproduction scheme. We have described the architecture that we have developed to mimic the routes that are learnt by the robot. This includes:

  - Path planning: dynamically computing a path towards the robot's position to the first point of a route.
  - Dynamic obstacle avoidance: avoids humans walking near the robot to safely reproduce a route.

F) A novel strategy based on reinforcement to learn the controller of a robot. This strategy allows:

  - Fast robot learning from scrach.
  - The creation of a unique controller suited to the environment where the robot operates.
  - A continuous adaptation to the changes of the environment where the robot operates.
  - Anyone can modify the behaviour of the robot regardless of his knowledge of robotics.

  Our proposal is based on reinforcement learning techniques. In this sense anyone with a wireless joystick connected to our robot is able to send reinforcements to the robot when he thinks that it is behaving incorrectly. In this way, our robot will learn how to perform the task that the human is teaching to him.

The behaviours and algorithms that have been proposed within this thesis can go beyond the scope of a general purpose tour-guide robot. In the coming years, robots are called to populate new environments like our homes of offices. Therefore, those new generations of robots will need to be aware of humans, interact with them, follow them, and cooperate with them, just like our tour-guide robot. For this reason, we believe that what has been proposed within this thesis (in part or as a whole) can be the basis for future robot behaviours that might even reach the market in the coming years.

An example of a different application of our person following behaviour to learn and reproduce routes is an intelligent wheelchair. They can be used to bring a patient to a doctor's room by letting the chair follow a nurse, and then autonomously send it back to the patient's room, using the path that the chair learnt while following the nurse.

Individual elements are also of great interest in other applications. An augmented reality graphical user interface like the one we have designed is quite useful to explain what a robot is sensing, and what he expects from humans. It can be applied to any other type of robot. The robust multi-sensor localization algorithm is also of great interest in any other robot that

works in indoor environments, but it also could be adapted to work on robot-like devices such as the widely extended smartphones nowadays. These devices are populated with many robot-like sensors, and could greatly benefit from a precise indoor positioning system.

Finally, in order to test our algorithms, we have showcased and tested our robot many times both inside and outside research environments. We went to many primary and secondary schools, but also people of all ages and from different backgrounds came to our research centre to get in touch with our robot. Most people could test themselves the behaviours of the robot, as well as, learn a bit of how it works.

7.1 Future work

The challenges of tour-guide robots has been studied over the, almost, past 15 years. The different proposals have evolved from ad-hoc solutions to a search for general and flexible strategies. In this regard, this thesis has contributed in creating adaptable solutions to many of the challenges presented in these robots. However, there is still way for improvement in several aspects of our robot:

- The human detector that we have developed is not able to correctly segment humans that are touching each other. An intelligent analysis of the human shape might help on this task, taking care that it does not compromise the efficiency of the algorithm.
- The human recognition in our robot maintains a unique model of each human that changes dynamically with time. Even tough it is good enough for most situations it could benefit from strategies that maintain several models of the same human.
- The human robot-interaction scheme should provide a way to automatically identify when a hand has been lost during its tracking, and recover from it. Right now, the user is responsible for maintaining his hands visible for the robot and correct possible losses of the hand.
- The gesture recognition could also be improved by making the robot able to learn and adapt his gesture recognition using those gestures performed by its users.
- In situations where the noise level allows speech recognition to work, it can be used in combination with gesture interaction.

- More sensors can be introduced in our multi-sensor localization algorithm, for example the new Bluetooth low energy (BLE) could be a good choice to help the robot locate itself.

- The laser model of the localization algorithm should be adapted to work in situations where many people is occluding it. A good approach would be to detect which laser beams interfere with people's legs and to remove them from the sensor model.

- The processing of voice messages when recording a route can also be greatly improved. A post-processing of the messages should be done to remove ambient noise. Moreover, the robot should allow to easily discard and record again a voice message.

- The route reproduction that we have designed is focused on accurately following the recorded path, without making sure that the human is following the robot and paying attention to the route that is being showcased. In this regard, if we change the pan motor of the RGB-D camera for one able to rotate 360 degrees, our robot could keep track of the human who has required a route.

Finally, the skills I got during the research that I had to develop for this thesis may be helpful for my new role as entrepreneur, since we just started a new technology-based company: "Situm Technologies SL".

Derived works

Journal publications

– V. Alvarez-Santos, A. Canedo-Rodriguez, R. Iglesias, X.M. Pardo, C.V. Regueiro, M. Fernandez-Delgado.
Route learning and reproduction in a tour-guide robot.
Robotics and Autonomous Systems, available online, 2014. DOI: 10.1016/j.robot.2014.07.013

– V. Alvarez-Santos and R. Iglesias and X.M. Pardo and C.V. Regueiro and A. Canedo-Rodriguez.
Gesture-based interaction with voice feedback for a tour-guide robot.
Journal of Visual Communication and Image Representation, no. 2, vol 13, pp. 499–509, 2014.

– V. Alvarez-Santos and X.M. Pardo and R. Iglesias and A. Canedo-Rodriguez and C.V Regueiro.
Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot.
Robotics and Autonomous Systems, no. 8, vol. 60, pp. 1021–1036, 2012.

– A. Canedo-Rodriguez and R. Iglesias and C.V. Regueiro and V. Alvarez-Santos and X.M. Pardo.
Self-organized multi-camera network for a fast and easy deployment of ubiquitous robots in unknown environments.
Sensors, no. 1, vol 13, p. 426, 2013.

– A. Canedo-Rodriguez and C.V. Regueiro and R. Iglesias and V. Alvarez-Santos and X.M.

Self-organized multi-camera network for ubiquituous robot deployment in unknown environments.
Robotics and Autonomous Systems, 2012.

– A. Canedo-Rodriguez and V. Alvarez-Santos and C.V. Regueiro and X.M. Pardo and R. Iglesias
Multi-agent system for fast deployment of a guide robot in unknown environments.
Special Issue on Advances on Physical Agents. In Journal of Physical Agents, vol. 6 no. 1, pp. 31-41, 2012.

International conferences

– V. Alvarez-Santos and A. Canedo-Rodriguez and R. Iglesias and X.M. Pardo and C.V. Regueiro
Route learning and reproduction in a tour-guide robot.
Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2013, vol. 6686 pp. 112-121, 2013.

– A. Canedo-Rodriguez and V. Alvarez-Santos and D. Santos-Saavedra and C. Gamallo and M. Fernandez-Delgado, R. Iglesias, C.V. Regueiro.
Robust multi-sensor system for mobile robot localization.
Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2013, vol. 6686 pp. pp. 112-121, 2013.

– V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, C. V. Regueiro.
Online Feature Weighting for Human Discrimination in a Person Following Robot.
Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2011, vol. 6686, pp. 222-232, 2011.

– A. Canedo-Rodriguez, R. Iglesias, C. V. Regueiro, V. Alvarez-Santos, X. M. Pardo.
Self-organized Multi-agent System for Robot Deployment in Unknown Environments.
Foundations on Natural and Artificial Computation, Lecture Notes in Computer Science, Proceedings of the IWINAC 2011, vol. 6686, pp. 165-174, 2011.

Spanish conferences

– V. Álvarez-Santos and R. Iglesias and X. M. Pardo and C. V. Regueiro and A. Canedo-Rodríguez.
Gesture based interface with voice feedback for a guide robot.
In Proceedings of the XIII Workshop of Physical Agents (WAF2012), pp. 135-144, 2012.

– David Santos-Saavedra, Xosé M. Pardo, Roberto Iglesias, Víctor Álvarez-Santos, Adrián Canedo-Rodríguez and Carlos V. Regueiro.
Global image features for scene recognition invariant to symmetrical reflections in robotics.
In Proceedings of the XV Workshop of Physical Agents (WAF2014), pp. 29-37, 2014.

– A. Canedo-Rodríguez, D. Santos-Saavedra, V. Alvarez-Santos, C. V. Regueiro, R. Iglesias and X.M. Pardo.
Analysis of different localization systems suitable for a fast and easy deployment of robots in diverse environments.
In Proceedings of the XIII Workshop of Physical Agents (WAF2012), pp. 39-46, 2012.

– A. Canedo-Rodriguez and V. Alvarez-Santos and C.V. Regueiro and X.M. Pardo and R. Iglesias.
Multi-agent system for fast deployment of a guide robot in unknown environments.
In Proceedings of the XII Workshop of Physical Agents (WAF2011), pages 23-30, 2011

Bibliography

[1] C. Balaguer, A. Barrientos, P. Sanz, R. Sanz, and E. Zalama, "Libro blanco de la robotica," *De la investigación al desarrollo y futuras aplicaciones. CEA-GTROB subencionado por el MEC. Espana*, 2007.

[2] J. Buhmann, W. Burgard, A. B. Cremers, D. Fox, T. Hofmann, F. E. Schneider, J. Strikos, and S. Thrun, "The mobile robot rhino," *AI Magazine*, vol. 16, no. 2, p. 31, 1995.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *AAAI/IAAI*, pp. 11–18, 1998.

[4] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial intelligence*, vol. 114, no. 1, pp. 3–55, 1999.

[5] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, *et al.*, "Minerva: A second-generation museum tour-guide robot," in *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*, vol. 3, IEEE, 1999.

[6] R. Siegwart, K. Arras, B. Jensen, R. Philippsen, and N. Tomatis, "Design, implementation and exploitation of a new fully autonomous tour guide robot," in *Proc. of the 1st Int. Workshop on Advances in Service Robotics (ASER)*, 2003.

[7] G. Kim, W. Chung, K.-R. Kim, M. Kim, S. Han, and R. H. Shinn, "The autonomous tour-guide robot jinny," in *Intelligent Robots and Systems, 2004.(IROS 2004).*

Proceedings. 2004 IEEE/RSJ International Conference on, vol. 4, pp. 3450–3455, IEEE, 2004.

[8] Y. Koide, T. Kanda, Y. Sumi, K. Kogure, and H. Ishiguro, "An approach to integrating an interactive guide robot with ubiquitous sensors," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2500–2505, IEEE, 2004.

[9] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, and S. Behnke, "Towards a humanoid museum guide robot that interacts with multiple persons," in *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pp. 418–423, IEEE, 2005.

[10] Y. Kuno, K. Sadazuka, M. Kawashima, K. Yamazaki, A. Yamazaki, and H. Kuzuoka, "Museum guide robot based on sociological interaction analysis," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1191–1194, ACM, 2007.

[11] T. Germa, F. Lerasle, P. Danes, and L. Brethes, "Human/robot visual interaction for a tour-guide robot," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3448–3453, IEEE, 2007.

[12] D. Rodriguez-Losada, F. Matia, R. Galan, M. Hernando, J. M. Montero, and J. M. Lucas, "Urbano, an interactive mobile tour-guide robot," *Advances in Service Robotics, Ed. H. Seok. In-Teh*, pp. 229–252, 2008.

[13] A. Chella and I. Macaluso, "The perception loop in cicerobot, a museum guide robot," *Neurocomputing*, vol. 72, no. 4, pp. 760–766, 2009.

[14] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita, "An affective guide robot in a shopping mall," in *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pp. 173–180, ACM, 2009.

[15] Bluebotics, "Gilberto â tour guide."

[16] R. Tellez, F. Ferro, S. Garcia, E. Gomez, E. Jorge, D. Mora, D. Pinyol, J. Oliver, O. Torres, J. Velazquez, *et al.*, "Reem-b: An autonomous lightweight human-size humanoid robot," in *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pp. 462–468, IEEE, 2008.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, 2009.

[18] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro, "Online feature weighting for human discrimination in a person following robot," in *Foundations on Natural and Artificial Computation*, pp. 222–232, Springer Berlin Heidelberg, 2011.

[19] V. Alvarez-Santos, X. M. Pardo, R. Iglesias, A. Canedo-Rodriguez, and C. V. Regueiro, "Feature analysis for human recognition and discrimination: Application to a person-following behaviour in a mobile robot," *Robotics and Autonomous Systems*, vol. 60, no. 8, pp. 1021–1036, 2012.

[20] S. Feyrer and A. Zell, "Detection, tracking, and pursuit of humans with an autonomous mobile robot," in *Intelligent Robots and Systems, 1999. IROS'99. Proceedings. 1999 IEEE/RSJ International Conference on*, vol. 2, pp. 864–869, IEEE, 1999.

[21] S. S. Ghidary, Y. Nakata, T. Takamori, and M. Hattori, "Human detection and localization at indoor environment by home robot," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 2, pp. 1360–1365, IEEE, 2000.

[22] L. Li, Y. T. Koh, S. S. Ge, and W. Huang, "Stereo-based human detection for mobile service robots," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, vol. 1, pp. 74–79, IEEE, 2004.

[23] R. Luo, Y. Chen, C. Liao, and A. Tsai, "Mobile robot based human detection and tracking using range and intensity data fusion," in *Advanced Robotics and Its Social Impacts, 2007. ARSO 2007. IEEE Workshop on*, pp. 1–6, IEEE, 2007.

[24] C. Martin, E. Schaffernicht, A. Scheidig, and H.-M. Gross, "Multi-modal sensor fusion using a probabilistic aggregation scheme for people detection and tracking," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 721–728, 2006.

[25] N. Bellotto and H. Hu, "Multimodal people tracking and identification for service robots," *International Journal of Information Acquisition*, vol. 5, no. 03, pp. 209–221, 2008.

[26] M. Mucientes and W. Burgard, "Multiple hypothesis tracking of clusters of people," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 692–697, IEEE, 2006.

[27] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1710–1715, IEEE, 2008.

[28] A. Treptow, G. Cielniak, and T. Duckett, "Real-time people tracking for mobile robots using thermal vision," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 729–739, 2006.

[29] A. Fernández-Caballero, J. C. Castillo, J. Martínez-Cantos, and R. Martínez-Tomás, "Optical flow or image subtraction in human detection from infrared camera on mobile robot," *Robotics and Autonomous Systems*, vol. 58, no. 12, pp. 1273–1281, 2010.

[30] C. Wong, D. Kortenkamp, and M. Speich, "A mobile robot that recognizes people," in *Tools with Artificial Intelligence, 1995. Proceedings., Seventh International Conference on*, pp. 346–353, IEEE, 1995.

[31] S. Chen, T. Zhang, C. Zhang, and Y. Cheng, "A real-time face detection and recognition system for a mobile robot in a complex background," *Artificial Life and Robotics*, vol. 15, no. 4, pp. 439–443, 2010.

[32] H. Zhou, M. Taj, and A. Cavallaro, "Target detection and tracking with heterogeneous sensors," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 4, pp. 503–513, 2008.

[33] G. Cielniak, T. Duckett, and A. J. Lilienthal, "Data association and occlusion handling for vision-based people tracking by mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 435–443, 2010.

[34] V. Alvarez-Santos, R. Iglesias, X. Pardo, C. Regueiro, and A. Canedo-Rodriguez, "Gesture-based interaction with voice feedback for a tour-guide robot," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 499–509, 2014.

[35] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 685–692, IEEE, 2010.

[36] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 10, pp. 1631–1643, 2005.

[37] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[38] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[39] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[40] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'reilly, 2008.

[41] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 17–24, ACM, 2007.

[42] H. Zhou, P. Miller, and J. Zhang, "Age classification using radon transform and entropy based scaling svm.," in *BMVC*, pp. 1–12, 2011.

[43] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using sift features and mean shift," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 345–352, 2009.

[44] D. Gossow, P. Decker, and D. Paulus, "An evaluation of open source surf implementations," in *RoboCup 2010: Robot Soccer World Cup XIV*, pp. 169–179, Springer, 2011.

[45] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[46] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Computer VisionâECCV'94*, pp. 151–158, Springer, 1994.

[47] M. Heikkilä, M. Pietikäinen, and C. Schmid, "Description of interest regions with center-symmetric local binary patterns," in *Computer Vision, Graphics and Image Processing*, pp. 58–69, Springer, 2006.

[48] Y. Mu, S. Yan, Y. Liu, T. Huang, and B. Zhou, "Discriminative local binary patterns for human detection in personal album," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.

[49] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[50] C. S. Won, D. K. Park, and S.-J. Park, "Efficient use of mpeg-7 edge histogram descriptor," *Etri Journal*, vol. 24, no. 1, pp. 23–30, 2002.

[51] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[52] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 8, pp. 1226–1238, 2005.

[53] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[54] C. van Rijsbergen, *Information Retrieval. 1979*. Butterworth, 1979.

[55] V. S. N. Prasad, A. Faheema, and S. Rakshit, "Feature selection in example-based image retrieval systems.," in *ICVGIP*, Citeseer, 2002.

[56] C. Granata, M. Chetouani, A. Tapus, P. Bidaud, and V. Dupourqué, "Voice and graphical-based interfaces for interaction with a robot dedicated to elderly and people with cognitive disorders," in *RO-MAN, 2010 IEEE*, pp. 785–790, IEEE, 2010.

[57] O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher, and R. Dillmann, "Using gesture and speech control for commanding a robot assistant," in *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on*, pp. 454–459, IEEE, 2002.

[58] K. Ishii, S. Zhao, M. Inami, T. Igarashi, and M. Imai, "Designing laser gesture interface for robot control," in *Human-Computer Interaction–INTERACT 2009*, pp. 479–492, Springer, 2009.

[59] Y. Okuno, T. Kanda, M. Imai, H. Ishiguro, and N. Hagita, "Providing route directions: design of robot's utterance, gesture, and timing," in *Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on*, pp. 53–60, IEEE, 2009.

[60] A. van Breemen, X. Yan, and B. Meerbeek, "icat: an animated user-interface robot with personality," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 143–144, ACM, 2005.

[61] S. Waldherr, R. Romero, and S. Thrun, "A gesture based interface for human-robot interaction," *Autonomous Robots*, vol. 9, no. 2, pp. 151–173, 2000.

[62] X. Liu and K. Fujimura, "Hand gesture recognition using depth data," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pp. 529–534, IEEE, 2004.

[63] M. Roccetti, G. Marfia, and A. Semeraro, "Playing into the wild: A gesture-based interface for gaming in public spaces," *Journal of Visual Communication and Image Representation*, vol. 23, no. 3, pp. 426–440, 2012.

[64] J. Gast, A. Bannat, T. Rehrl, F. Wallhoff, G. Rigoll, C. Wendt, S. Schmidt, M. Popp, and B. Farber, "Real-time framework for multimodal human-robot interaction," in *Human System Interactions, 2009. HSI'09. 2nd Conference on*, pp. 276–283, IEEE, 2009.

[65] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Design of an office-guide robot for social interaction studies," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 4965–4970, IEEE, 2006.

[66] H.-M. Gross, H. Boehme, C. Schröter, S. Mueller, A. Koenig, E. Einhorn, C. Martin, M. Merten, and A. Bley, "Toomas: interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 2005–2012, IEEE, 2009.

[67] K. Yelamarthi, S. Sherbrook, J. Beckwith, M. Williams, and R. Lefief, "An rfid based autonomous indoor tour guide robot," in *IEEE 55th International Midwest Symposium on Circuits and Systems, MWSCAS, August*, pp. 5–8, 2012.

[68] R. Stricker, S. Muller, E. Einhorn, C. Schroter, M. Volkhardt, K. Debes, and H.-M. Gross, "Interactive mobile robots guiding visitors in a university building," in *RO-MAN, 2012 IEEE*, pp. 695–700, IEEE, 2012.

[69] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent asimo: System overview and integration," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, pp. 2478–2483, IEEE, 2002.

[70] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 41, no. 3, pp. 569–573, 2011.

[71] A. Just and S. Marcel, "A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition," *Computer Vision and Image Understanding*, vol. 113, no. 4, pp. 532–543, 2009.

[72] P. Hong, M. Turk, and T. S. Huang, "Gesture modeling and recognition using finite state machines," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pp. 410–415, IEEE, 2000.

[73] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis, "A hidden markov model-based continuous gesture recognition system for hand motion trajectory," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[74] M. Moni and A. S. Ali, "Hmm based hand gesture recognition: A review on techniques and approaches," in *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 433–437, IEEE, 2009.

[75] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 26, no. 1, pp. 43–49, 1978.

[76] H. Huttenrauch, A. Green, M. Norman, L. Oestreicher, and K. S. Eklundh, "Involving users in the design of a mobile office robot," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 2, pp. 113–124, 2004.

[77] B. Jensen, G. Froidevaux, X. Greppin, A. Lorotte, L. Mayor, M. Meisser, G. Ramel, and R. Siegwart, "The interactive autonomous mobile system robox," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2, pp. 1221–1227, IEEE, 2002.

[78] F. Faber, M. Bennewitz, C. Eppner, A. Gorog, C. Gonsior, D. Joho, M. Schreiber, and S. Behnke, "The humanoid museum tour guide robotinho," in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pp. 891–896, IEEE, 2009.

[79] H. Avilés, M. Alvarado-González, E. Venegas, C. Rascón, I. V. Meza, and L. Pineda, "Development of a tour–guide robot using dialogue models and a cognitive architecture," in *Advances in Artificial Intelligence–IBERAMIA 2010*, pp. 512–521, Springer, 2010.

[80] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, *et al.*, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *International Journal of Robotics Research*, vol. 19, no. 11, pp. 972–999, 2000.

[81] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data," *The International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[82] C. Gamallo, C. Regueiro, P. Quintía, and M. Mucientes, "Omnivision-based kld-monte carlo localization," *Robotics and Autonomous Systems*, vol. 58, no. 3, pp. 295–305, 2010.

[83] A. Canedo-Rodriguez, D. Santos-Saavedra, V. Alvarez-Santos, C. V. Regueiro, R. Iglesias, and X. M. Pardo, "Analysis of different localization systems suitable for a fast and easy deployment of robots in diverse environments," in *Workshop of Physical Agents*, pp. 39–46, 2012.

[84] A. Canedo-Rodriguez, V. Alvarez-Santos, D. Santos-Saavedra, C. Gamallo, M. Fernandez-Delgado, R. Iglesias, and C. V. Regueiro, "Robust multi-sensor system for mobile robot localization," in *Natural and Artificial Computation in Engineering and Medical Applications*, pp. 92–101, Springer Berlin Heidelberg, 2013.

[85] J.-S. Gutmann and D. Fox, "An experimental comparison of localization methods continued," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 1, pp. 454–459, IEEE, 2002.

[86] S. Thrun, W. Burgard, D. Fox, *et al.*, *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.

[87] D. Pollard, *A user's guide to measure theoretic probability*, vol. 8. Cambridge University Press, 2001.

[88] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 6, pp. 1067–1080, 2007.

[89] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 1, pp. 13–32, 2009.

[90] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[91] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[92] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[93] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm.," in *ICAPS*, pp. 262–271, 2005.

[94] N. A. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1617–1624, IEEE, 2007.

[95] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids.," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 22, p. 1177, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[96] M. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, vol. 9, no. 3, pp. 1102–1110, 2009.

[97] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Operations Research*, vol. 60, no. 3, pp. 611–624, 2012.

[98] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.

[99] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, p. 0278364913495721, 2013.

[100] A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[101] P. Quintia, R. Iglesias, M. A. Rodríguez, and C. V. Regueiro, "Simultaneous learning of perception and action in mobile robots," *Robotics and autonomous systems*, vol. 58, no. 12, pp. 1306–1315, 2010.

[102] P. Quintía Vidal, R. Iglesias Rodríguez, M. Á. Rodríguez González, and C. Vázquez Regueiro, "Learning on real robots from experience and simple user feedback," 2013.

[103] R. Iglesias Rodríguez, M. Á. Rodríguez González, P. Quintía Vidal, and C. Vázquez Regueiro, "Continuous learning on a real robot through user feedback," 2012.

[104] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural networks*, vol. 4, no. 6, pp. 759–771, 1991.

[105] M. Bozarth, "Pleasure: The politics and the reality," 1994.

[106] E. L. Thorndike, *Animal intelligence: Experimental studies*. Macmillan, 1911.

[107] A. L. Thomaz and C. Breazeal, "Teachable robots: Understanding human teaching behavior to build more effective robot learners," *Artificial Intelligence*, vol. 172, no. 6, pp. 716–737, 2008.

[108] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2169–2178, IEEE, 2006.

[109] K. Kristo and C.-S. Chua, "Image representation for object recognition: Utilizing overlapping windows in spatial pyramid matching.," in *ICIP*, pp. 3354–3357, 2013.

[110] A. J. Sharkey, *Combining artificial neural nets: ensemble and modular multi-net systems*. Springer-Verlag New York, Inc., 1999.

[111] M. Rodriguez, R. Iglesias, and P. Q. C. Regueiro, "Parallel robot learning through an ensemble of predictors able to forecast the time interval before a robot failure," in *XI Workshop of Physical Agents*, 2010.

List of Figures

List of Tables