Universidad
Carlos III de Madrid
www.uc3m.es

TESIS DOCTORAL

Arquitectura y diseño de un sistema completo de navegación semántica. Descripción de su ontología y gestión de conocimiento.

Autor:
Jonathan Crespo Herrero
Director:
Ramón I. Barber Castaño

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
Leganés, Fecha

TESIS DOCTORAL (DOCTORAL THESIS)

Arquitectura y diseño de un sistema completo de navegación semántica. Descripción de su ontología y gestión de conocimiento.

Autor (Candidate): Jonathan Crespo Herrero

Director (Adviser): Ramón I. Barber Castaño

Tribunal (Review Committee)

Firma (Signature)

Presidente (Chair): _____

Vocal (Member): _____

Secretario (Secretary): _____

Título (Degree): Doctorado en Ingeniería Eléctrica, Electrónica y Automática
Calificación (Grade): _____ Leganés, de de

A mi familia

"Podría parecer que hemos llegado a los límites alcanzables por la tecnología informática, aunque uno debe ser prudente con estas afirmaciones, pues tienden a sonar bastante tontas en cinco años."

— John Von Neumann, sobre 1949

Agradecimientos

Son muchas las personas a las que mencionar en este momento. Pero sin duda por quien debo empezar es por mi director de tesis, Ramón Barber. Por su dedicación, apoyo y confianza, sin los cuales esta tesis no habría sido posible. Sus sabios consejos, directrices y recomendaciones durante los años que he dedicado a este trabajo han sido invaluables, mostrándome siempre el camino correcto. Sus enseñanzas me serán de gran ayuda para todo mi futuro, trascendiendo de esta tesis.

Quiero extender los agradecimientos a todo el Robotics Lab de la Universidad Carlos III de Madrid, que me acogieron desde el primer momento como uno más de una gran familia dirigida por Carlos Balaguer, Miguel A. Salichs y Luis Moreno. Y a toda la gente que ha convivido conmigo en el despacho o en el laboratorio, pasando muchas horas con Fernando Alonso, Edwin Oña, Esther Salichs, Arnaud Ramey, Álvaro Castro y mucha más gente que han hecho únicos estos años, no puedo nombrar a todos, pero si estáis leyendo esto es porque va por vosotros. En especial quiero mencionar a José Carlos, porque es ese compañero que no duda en ayudarte cuando te hace falta y del que se puede aprender mucho en cuanto a conocimientos y actitud.

Hay más gente a la que debo agradecer. A Óscar Martínez Mozos, por tantos buenos consejos. A Michael Beetz, por haberme aceptado tres meses en su grupo de investigación en el Instituto de Inteligencia Artificial de la Universidad de Bremen. A Carlos, Íñigo, Enrique, Clara y Alejandra; que fueron mis estudiantes de máster a quienes pude codirigir sus respectivos trabajos que contribuyeron en varios aspectos a esta tesis.

Por supuesto a mis padres, porque son mi ejemplo de actitud ante la vida, por haberme enseñado a luchar y a no darme por vencido, por haberme dado siempre su apoyo incondicional, porque son los primeros en confiar en mí, por todos los valores que me han enseñado y por poder contar con ellos.

Y finalmente a Cristina, por demasiadas cosas como para enumerarlas, entre las que han estado sus ánimos y muestras de apoyo constantes.

A todos, ¡GRACIAS!

Resumen

En el actual mundo de la robótica, existe una tendencia general de implementar mecanismos conductuales basados en la psicología humana tomando como ejemplo el procesamiento natural del pensamiento. Esto permite una mayor comprensión del entorno y de los objetos que contiene. Esta tesis se desarrolla con la idea subyacente de conseguir que un robot se comporte de manera semejante a como lo haría un ser humano. Esto no es algo inusual, muchos investigadores dedican su esfuerzo a crear sistemas que resuelven una determinada tarea de una manera similar a como lo resolvería una persona. En el caso de esta tesis, la tarea o habilidad sobre la que se ha trabajado ha sido la navegación de robots móviles.

En el caso de la navegación es necesario que se procesen una serie de subtareas de cuya eficiencia va a depender que el robot llegue realmente al lugar al que quería ir. Se necesita un planificador que decida los lugares a los que se debe dirigir el robot. Y si esas decisiones se toman en base a una clasificación razonada del entorno, que se realiza en función de una serie de conceptos que definen el mundo que rodea al robot, se está realizando una navegación a un nivel más abstracto que el geométrico o topológico, más cercano a la manera de proceder de un ser humano. En esta tesis, este nivel de navegación es referido como navegación semántica, tal como otros autores han hecho anteriormente.

En este punto, se puede definir que el objetivo principal de esta tesis es diseñar e implementar un sistema de navegación semántica que gestione una mayor cantidad de conceptos y relaciones, al mismo tiempo que se estudian las implementaciones más eficientes. Esto permite clasificar de una manera precisa el entorno y dota de mayor autonomía al robot a la hora de encontrar rutas que le acerquen a su destino.

Habiendo enmarcado la tesis dentro de este objetivo, se ha trabajado sobre el diseño de una ontología para definir el entorno. Todos los navegadores necesitan manejar un mapa que es interpretado por el planificador y que además sirve para que el robot se localice en el mundo que le rodea. En el caso de la navegación semántica, la información representada en esta ontología funciona como mapa semántico y cumple con esta función. El diseño de la ontología también es determinante para el sistema de razonamiento que necesita incorporar el navegador. Este sistema de razonamiento es el que se encarga de extraer la información necesaria a partir de los datos de los que se dispone y que se almacenan como entidades de la ontología.

La navegación semántica requiere además que se incorporen al robot varios subsistemas que aporten distintas habilidades. La capacidad sensorial es una de estas habilidades. Debido a las características de este tipo de navegación, la capacidad sensorial es mucho más exigente que en otros navegadores puesto que el robot necesita percibir los objetos del entorno. Otra habilidad destacable es la capacidad de interacción con los usuarios humanos. Además, se necesita un sistema de navegación geométrico o topológico para el bajo nivel, ya que la navegación semántica ha sido tratada como una capa superior de abstracción que se construye sobre otro navegador.

En esta tesis se detalla cómo funciona el sistema navegador completo, el desarrollo de los subsistemas y la integración de todos ellos. La arquitectura del sistema ha sido diseñada para ser modular, con lo que se ha podido implementar varias soluciones diferentes para todos los módulos. Esto ha llevado a integrar varios sistemas de percepción, probar con distintos sistemas de razonamiento, dos tipos de navegadores (uno geométrico y otro topológico) y dos interfaces de usuario (una por teclado y otra basada en un sistema de diálogo con voz). Esto sugiere que el sistema de navegación desarrollado en esta tesis es a su vez una buena herramienta para analizar y comparar los subsistemas por separado.

La gestión de la información semántica es otra de las cuestiones relevantes en esta tesis. Por ello quedan descritos los procedimientos mediante los que se obtiene la información necesaria para el razonador. Se contempla la recursividad de destinos, puesto que la respuesta a una petición de destino puede ser desconocida, pero sucesivas iteraciones pueden llevar a un destino conocido. Además, se han estudiado varias maneras de ampliar la información que tiene el sistema. Esto es, por lo tanto, aprender. Las vías de aprendizaje contempladas, aparte de la introducción manual de datos, son la clasificación autónoma de estancias, consultas en webs de conocimiento y la interacción con humanos mediante diálogo. La capacidad de adquisición de nuevos conceptos o relaciones por parte del robot, está directamente relacionada con su autonomía. El robot no necesita de la intervención humana si es capaz de aprender nuevos conceptos por sí mismo.

Una vez explicados los sistemas desarrollados en esta tesis, se describen una serie de experimentos bien documentados. Se han incluido pruebas de integración, pruebas de subsistemas individuales, pruebas de aprendizaje y pruebas de razonamiento.

Abstract

In the current robotics, there is a general tendency to implement behavioral mechanisms based on human psychology, taking the natural processing of thought as an example. This provides a greater understanding of the environment and the contained objects. This thesis is developed with the underlying idea of getting a robot to behave in a similar way as a human would. This is not unusual, many researchers dedicate their effort to create systems that solve a certain task in a manner similar as a person solve it. The focused task or skill in this thesis is the navigation of mobile robots.

In the case of navigation, processing a series of subtasks is necessary. The subtask efficiency influences on the capability of the robot to reach the desired place. A planner is needed to decide where the robot should be go. In this thesis the navigation has a level of abstraction higher than in geometric or topological navigation. The decisions of the planner are made based on a reasoned classification of the environment. The classification is made according to the concepts that define the world that surrounds the robot. This navigation is closer to the way of a human being. In this thesis, this level of navigation is named semantic navigation, as other authors have named it before.

The main objective of this thesis is to design and implement a semantic navigation system that manages a greater number of concepts and relationships. In addition, the most efficient implementations have been studied. This allows a precise classification

of the environment. This also gives the robot greater autonomy to find routes that bring it closer to its destination.

The design of an ontology to define the environment is presented. All navigators need to manage a map that the planner interprets. In addition, the map is used for the robot to be located in the world around it. In the case of semantic navigation, the information represented in this ontology functions as a semantic map and fulfills these functions. The design of the ontology is determinant for the reasoning system that needs to incorporate the navigator. This system of reasoning is responsible for extracting the necessary information from the data available. The data are stored as entities of the ontology.

Semantic navigation requires that several subsystems that contribute different skills are incorporated into the robot. Sensory ability is one such skill. Due to the characteristics of this type of navigation, the sensorial capacity is much more demanding than in other navigation systems. This is because the robot needs to perceive the objects in the environment. Another remarkable skill is the ability to interact with human users. In addition, a geometric or topological navigation system is required for the low level, since semantic navigation has been treated as an upper layer of abstraction that is built on another navigation system.

This thesis details the complete operation of the navigation system, the development of the subsystems and the integration of all of them. The system architecture has been designed to be modular. For this reason it has been possible to implement several different solutions for all the modules. This has led to the integration of several perception systems, to test different systems of reasoning, two types of browsers (one geometric and one topological) and two user interfaces (one by keyboard and another based on a dialog system with voice). This suggests that the navigation system developed in this thesis is also a good tool to analyze and compare the subsystems separately.

The management of semantic information is another relevant issue in this thesis. Therefore, the procedures by which the necessary information for the reasoner is obtained are described. Recursion of destinations is contemplated, since the response to a destination request may be unknown, but successive iterations can lead to a known destination. In addition, several ways of acquiring new information have been studied.

The paths of learning contemplated are the manual introduction of data, the autonomous classification of stays, consultations in knowledge webs and the interaction with humans through dialogue.

Once the systems developed in this thesis have been explained, a set of well-documented experiments are described. Integration tests, individual subsystem tests, learning tests and reasoning tests have been included.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

"Los científicos estudian el mundo tal como es, los ingenieros crean el mundo que nunca ha sido." — Theodore von Karman

En el mundo de la robótica existen sistemas que incluyen la capacidad de trabajar con representaciones dotadas de significado a partir de percepciones del propio entorno, enfocando el robot a tareas y situaciones de la vida real. Un alto nivel de abstracción y que contemple gran cantidad de información sobre objetos y lugares, facilita a los sistemas de inferencia adquirir aún más información. Los robots que cuentan con arquitecturas cognitivas que modelan y emplean la semántica del entorno, aumentan su grado de autonomía además de ser más robustos y eficientes. En el área de la navegación de robots móviles esto significa dotar al robot de la capacidad de entender el entorno que le rodea de una manera similar a como lo hacen los seres humanos. La navegación semántica [67] aborda este tema, incorporando conceptos de la robótica cognitiva a la navegación.

La navegación semántica permite que el robot pueda asociar lo que percibe a los lugares en los que se encuentra, el sistema recoge y utiliza la información del entorno y establece relaciones entre los conceptos que lo definen. Hay muchos elementos del entorno que pueden ser utilizados para extraer conclusiones sobre el mismo. Es por

esto que el navegador desarrollado en esta tesis se apoya en un sistema de representación relacional del entorno, en el que una ontología describe las relaciones entre conceptos, los cuales forman parte de una jerarquía conceptual. Esta jerarquía conceptual se relaciona a su vez con una jerarquía espacial. En este sistema, las clases de estancias se relacionan con los objetos que hay en ellas mientras que los objetos se relacionan con su utilidad, características y con otros objetos. De esta manera, las estancias también se pueden asociar con utilidades. Por ejemplo, de un lugar que contiene objetos que se utilizan para trabajar, se deduce que es un buen destino para alguien que solicite que quiere trabajar. También se pueden realizar deducciones del tipo de que es posible encontrar en la cocina objetos como el frigorífico o un hornillo. Además, en otro ejemplo, como se ve en la figura 1.1, un bolígrafo puede estar asociado con el concepto de *escribir* y relacionado con *papel*. Pero *escribir* también puede estar asociado con *trabajar*. Y si también se relaciona con la utilidad *dibujar*, ésta a su vez puede estar relacionada con *ocio*. Y el *ocio* con *actividad relajante*. Todos estos conceptos se emplean para clasificar el lugar donde se encuentran los objetos y esa información se emplea para alcanzar una ubicación determinada para cumplir un objetivo concreto. El robot puede encontrar de esta forma lugares o estancias relacionadas semánticamente con el objetivo, aunque esté en un entorno poco explorado o desconocido. Este nivel de navegación permite completar la información necesaria mediante el conocimiento parcial del entorno.

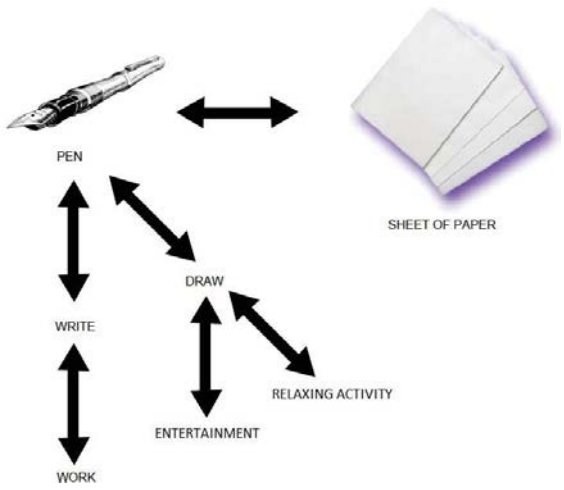


Figura 1.1: Ejemplo de relaciones semánticas entre objetos y conceptos

En esta tesis se muestra el funcionamiento de un módulo de razonamiento que sirve tanto para almacenar los objetos y la información percibida como para inferir nuevo conocimiento en base a las relaciones de las los objetos. Este módulo de razonamiento es uno de los pilares sobre los que se puede construir una navegación semántica. El diseño propuesto se ajusta a la ontología definida sobre el entorno en el que navegará el robot. El sistema de navegación consta de varios módulos que soportan las distintas habilidades necesarias para cumplir la tarea de la navegación. Estas habilidades son comentadas a lo largo del documento, mostrando cómo se incorpora cada una en el capitulo 3, destinado a describir el sistema de navegación completo.

1.1. Motivación

Esta tesis se ocupa de la navegación semántica y de las mejoras que ofrece respecto a otros sistemas de navegación. Se ha decidido investigar en los recursos que dotan de autonomía al robot y le permiten la toma de decisiones. Esto hace que un punto importante a estudiar sea la habilidad de razonamiento que presentan este tipo de navegadores y la robustez ante fallos o ausencias de información que ofrecen.

Además se ha considerado apropiado investigar los medios de adquisición de nuevo conocimiento para aportar alternativas y complementar unos sistemas con otros. El mapa semántico se forma con los conceptos y la información sobre los objetos y sus relaciones entre ellos y el entorno. Por esto se han expuesto los mecanismos empleados para añadir nuevo conocimiento. Esto ha llevado a su vez a dedicar parte del esfuerzo a integrar distintos sistemas de percepción en el robot, lo que aumenta a la vez la información que contienen los mapas semánticos y la capacidad de deducir relaciones en el entorno.

En esta tesis se presenta la arquitectura completa de un navegador semántico que permite la integración de diferentes soluciones a las distintas tareas que debe realizar. Resulta interesante la posibilidad de poner a prueba cada uno de los módulos involucrados. De este modo se pueden emplear navegadores de bajo nivel diferentes, distintos sistemas de percepción y diferentes subsistemas de comunicación con el usuario.

1.2. Objetivos

La navegación semántica es actualmente el nivel más alto de abstracción al que puede navegar un robot. Otorga un desplazamiento autónomo, robusto ante imprevistos y eficiente. Los objetivos de esta tesis se enmarcan dentro de la siguiente meta propuesta:

"Construir un sistema de navegación para un robot móvil con la capacidad de adquirir y gestionar la información semántica del entorno"

Para conseguir esta meta, se definen los siguientes objetivos secundarios:

- Modelado semántico del entorno. Se pretende estudiar las distintas alternativas de modelado y desarrollar una propuesta que satisfaga adecuadamente las necesidades del sistema de navegación.
- Gestión de la información semántica, presentando un sistema de inferencia que dote al robot de mayor autonomía.
- Aprendizaje y adquisición de información semántica, incluyendo la clasificación de lugares.
- Desarrollo de cada componente necesario para el funcionamiento del sistema de navegación, incorporando información semántica cuando sea posible.
- Integración de todos los subsistemas desarrollados, permitiendo la integración presente o futura con distintas implementaciones de un mismo subsistema.

1.3. Contribuciones

El trabajo realizado supone una serie de aportes respecto al estado del arte:

- El modelo relacional mediante el que se describe la ontología es implementado con una base de datos y un software de gestión de la información semántica. Esto representa una alternativa que ofrece la posibilidad de trabajar con navegadores

semánticos sin tener que emplear sistemas de razonamiento basados en reglas. Además, esta alternativa requiere menor esfuerzo computacional.

- La ontología contempla relaciones y conceptos no utilizados en otros modelos, como el concepto de los significados subjetivos o el hecho de que las características de un objeto pueden modificar su localización. Los sistemas clásicos se reducen a establecer relaciones de objetos con estancias o con otros objetos, pero ignoran la importancia de las acciones que se pueden realizar con los objetos y la clasificación de las estancias en función a estas acciones.

- La integración de todos los componentes con identidad propia que constituyen el navegador es un aporte por sí mismo, sin embargo se añade el hecho de que al permitir y facilitar la sustitución por otros sistemas diferentes, se ha aportado una herramienta muy útil para comparar las mejoras individuales que ofrece un subsistema.

Otros aportes más indirectos han sido:

- La inclusión de técnicas de clasificación de estancias basadas en lo que realizan las personas que se encuentran en el entorno, puesto que en el estado del arte las clasificaciones se realizan únicamente en función de las características de la propia estancia y de los objetos que contiene.

- La agregación multi-modal de percepciones de objetos por distintas técnicas.

.

1.4. Estructura del documento

Esta tesis está estructurada con la siguiente división de capítulos, aparte de la presente introducción:

- El capítulo 2 recoge los aspectos más importantes del estado del arte actual sobre las líneas de investigación que convergen en la navegación semántica.

- El capítulo 3 describe la estructura y los elementos que componen el navegador semántico desarrollado.
- El capítulo 4 aborda los sistemas cognitivos empleados para gestionar la información semántica.
- El capítulo 5 se centra en las distintas maneras de adquirir nuevo conocimiento que se han incorporado.
- El capítulo 6 analiza los resultados experimentales de las pruebas y ensayos realizados.
- El capítulo 7 presenta las conclusiones y los aportes realizados respecto a lo existente en el estado del arte. Además se comenta el trabajo que genera esta tesis como proyectos futuros.

CAPÍTULO 2

Estado del Arte

"Investigar es ver lo que todo el mundo ha visto, y pensar lo que nadie más ha pensado." — Albert Szent-Györgi

En los últimos años hay un creciente interés en la dotación de habilidades cognitivas de alto nivel en todas las aplicaciones de la robótica. Los robots con capacidades de aprendizaje y razonamiento se suponen más eficientes y con mayor capacidad de reacción ante imprevistos. En el área de la robótica móvil también se aprecia la tendencia de elevar el nivel de abstracción en las tareas de navegación, habiendo cada vez más investigaciones sobre navegación semántica. La aplicación de conocimiento semántico para la navegación de robots móviles, representa un gran avance respecto a navegadores que únicamente pueden emplear mapas métricos [5] [24], topológicos [26] [131], o híbridos (combinación de los dos anteriores) [134] [137]. Los sistemas semánticos se consideran mejor adaptados para la interacción con humanos. En [66], aplican estos conceptos para entender las interacciones del lenguaje natural cuando una persona solicita a un robot encontrar un objeto nuevo y el robot debe buscar en el entorno ese objeto.

El primer asunto a abordar es la manera de representar el conocimiento. Las ontologías se pueden considerar como una de las mejores formas de obtener esta

representación [52]. Se pueden establecer jerarquías donde los niveles más altos de abstracción definan su interpretación semántica. Este conocimiento es considerado para la creación de mapas semánticos.

2.1. Representación del conocimiento

El conocimiento puede ser representado de distintas maneras. A mayor nivel de abstracción conceptual, se encuentran mayores posibilidades de tratar la información. En [101] muestran una visualización de la representación relacional de los datos en la figura 2.1. Describen conocmiento de sentido común en forma de relaciones entre conceptos (por ejemplo, la cocina *tiene-el-objeto* cereales), y describen las instancias de conocimiento como relaciones entre instancias y conceptos (por ejemplo, el objeto-1 *es-una-instancia-de* cereales), o instancias y otras instancias (el lugar X *tiene-el-objeto* objeto Y). Las relaciones en el mapa conceptual están predefinidas, pueden ser adquiridas o inferidas y pueden ser deterministas o probabilísticas (las relaciones son probabilísticas cuando se introduce incertidumbre). Sin embargo, aunque la capacidad de inferencia es muy alto y contemplan múltiples variables, está más limitado que el sistema de aprendizaje de esta tesis doctoral. En ese trabajo la inferencia se reduce a deducciones sobre el espacio inexplorado como predecir la presencia de objetos en una ubicación con categoría conocida. También pueden predecir la existencia de una estancia perteneciente a una categoría dentro de un lugar inexplorado. En el trabajo de esta tesis no sólo se pueden hacer esas predicciones, sino que además se ha potenciado las características particulares de algunos tipos de objeto, aportando mayor diversidad. Además, se contemplan también características del entorno que no se tenían en cuenta en este trabajo, como por ejemplo el nivel sonoro y el número de personas transitando cada estancia. Y se ha cubierto huecos que los autores dejaban para el futuro en sus conclusiones, como el problema del aprendizaje de nuevas propiedades y categorías de habitaciones de forma autónoma por el robot, encaminándose a un mapeo semántico auto-extensible. Los mecanismos de aprendizaje de esta tesis permiten solucionar ese problema, entre otros.

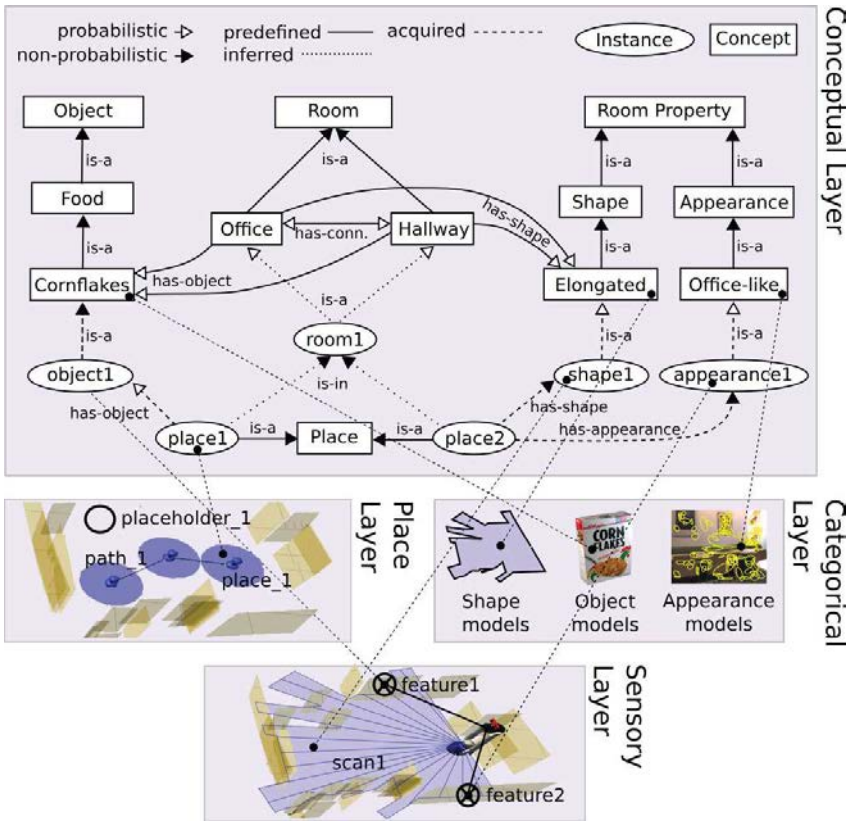


Figura 2.1: Estructura capeada de la representación espacial. La capa conceptual comprende conocmiento sobre conceptos, relaciones entre los conceptos e instancias de entidades espaciales.

Es interesante estudiar cómo se enfrentan otros autores a este problema. En [113] presentan una novedosa representación de un mapa semántico al que llaman *Multi-versal Semantic Map*, con el que pretenden extender el trabajo de Galindo et al. [44] y consideran las distintas combinaciones de posibles bases o universos, como instancias de ontologías [140] con anotaciones de creencia (certidumbre) sobre los conceptos y relaciones que sirven de fundamentos.

El trabajo de Galindo et al. es uno de los primeros en este campo, donde se presentan modelos de representación multi-jerárquicos y eligen NeoClassic como herramienta de representación de conocimiento y mecanismo de inferencia mediante *Lógicas Descriptivas* (DL). El sistema de clasificación está limitado a contornos, empleando cajas o cilindros como muebles. El potencial de esta representación fue explorado en

posteriores trabajos como en [43] donde mejoran el planificador y en [45] donde incluyen una generación autónoma de objetivos. En la misma línea se encuentran trabajos de otros autores como Zender et al. [157], donde la representación multi-jerárquica se reemplaza por una jerarquía simple trasladando los datos de mapas construidos a partir de los sensores a abstracciones conceptuales. Esta información es codificada mediante OWL-DL (*Ontology Web Language - Description Logic*), resultando en una ontología que define un dominio de oficina.

Otros autores como Nüchter y Hertzberg [92] emplean Prolog para implementar redes de restricciones con las que codifican las propiedades y relaciones entre las distintas superficies planas de un edificio (pared, suelo, techo y puerta) mientras que para la clasificación de objetos implementan dos técnicas, una basada en contornos y otra en una cascada de clasificadores que emplean datos de distancias y reflectancias. También se pueden encontrar otros sistemas de adquisición, representación y uso de mapas semánticos, como el denominado KnowRob-Map por Tenorth et al. [133], donde emplean redes de lógica Bayesiana para predecir la localización de objetos según sus relaciones. Todo ello implementado en SWI-Prolog y utilizando una ontología OWL-DL.

Estos trabajos establecen un claro camino hacia la utilización de ontologías para codificar el conocimiento semántico

2.1.1. Ontologías

En el ámbito de las ciencias de la computación, las ontologías son herramientas formales que permiten describir objetos, propiedades y relaciones de los mismos en un dominio de conocimiento. Según Prestes et al., en la literatura se encuentran dos definiciones en concreto que capturan la esencia del propósito y el alcance de una ontología. En [128] aportan una definición que es el resultado de combinar dos definiciones propuestas por Gruber y Borst y exponen que una ontología es "*una especificación formal explícita de una conceptualización compartida*". La otra definición vienen dada por Guarino enunciada como "*teorías lógicas que explican lo que intenta transmitir un vocabulario*". Una ontología comprende un conjunto de términos y sus definiciones como compartidos por una comunidad dada, formalmente especificada en un lenguaje legible por una máquina, como puede ser una lógica de primer orden.

Los elementos principales de una ontología son [51]

- **Clases**. Representan los conceptos a todos los niveles.
- **Relaciones**. Representan las asociaciones entre conceptos.
- **Axiomas formales**. Son restricciones o reglas que aportan consistencia a las estructuras de las relaciones.

Aunque las ontologías tienen en común los elementos principales que se acaban de describir, existen distintas maneras para clasificar y diferenciar las ontologías. En [100] eligen el nivel de generalidad para realizar la siguiente clasificación de ontologías, basándose en [53]:

- **Ontologías de alto nivel**. Describen conceptos muy generales, como espacio, tiempo, materia, objetos, eventos, acciones, etc. Esta generalidad les hace independientes de un problema o dominio en particular.
- **Ontologías de dominio**. Este tipo describe conceptos de un determinado dominio, orientados a resolver distintos problemas siempre y cuando se encuentren en un dominio concreto y acotado.
- **Ontologías de tarea**. Describen tareas o actividades genéricas.
- **Ontología de aplicación**. Están estrictamente relacionadas con una aplicación específica y son usadas para describir conceptos de un dominio y tarea particular.

En esta tesis, la ontología desarrollada corresponde a una ontología de tarea, puesto que resuelve la navegación en un entorno con gran generalidad. Aunque por sus características, queda cerca de una ontología de alto nivel.

Mientras que en el campo de la robótica y la automatización, se han desarrollado ontologías de dominio. En [73] se presenta una ontología OWL (*Lenguaje de Ontologías Web*) para describir el hardware, software y habilidades de un robot. El motor de razonamiento puede deducir qué habilidades faltan para completar una determinada tarea. En [132] representan los conceptos relativos al dominio de la robótica y los

entornos domésticos para proporcionar el correspondiente vocabulario mientras que en [156] desarrollan un modelo ontológico para robots industriales llamado *ReApp*. Otros ejemplos de ontologías son las empleadas en [116], donde buscan un estándar internacinal con ORA (*Ontology for Robotics and Automation*) y las ontologías de dominio usadas en [60] donde explotan la semántica implícita contenida en AML, definiendo las relaciones entre conceptos AML (*Automation Markup Language*) y el dominio terminológico de conocimiento codificado en las ontologías.

2.2. Empleo del conocimiento semántico

En el caso de la navegación, el conocimiento semántico se emplea principalmente en dos tareas concretas. Una es en el proceso de obtener un mapa del entorno. Esta tarea es crucial, puesto que es cuando se extrae la información del entorno y se plasma en algún tipo de mapa que el robot pueda entender. La otra tarea son los razonamientos e inferencias que el robot realiza a partir de ese conocimiento, que se puede ver en la robótica cognitiva.

2.2.1. Mapeo semántico

En la tarea de navegación de robots móviles, es fundamental definir claramente la manera de mapear el entorno. El tipo de mapa depende del nivel de abstracción en el que trabaje el navegador, pues la información que requiere cada tipo de navegación es diferente. En un navegador geométrico [48], la información del mapeado necesaria corresponde a distancias que mide el robot. En ese caso, el mapa tiene como objetivo determinar las zonas del espacio que corresponden a obstáculos y las que son espacio libre transitable. Y para ello emplea la información de los sensores de distancia, que indican en cada posición el espacio libre que tiene el robot en su alrededor. En el caso de los navegadores de nivel topológico [96], la información que requieren del entorno es la de conectividad entre áreas. Eso permite construir árboles de nodos que representan lugares del espacio y poder calcular rutas de un nodo a otro. Trabajos como [101] realizan lo que ellos llaman mapeo semántico. Consideran aplicaciones donde el robot se mueve en entornos domésticos o de oficinas, por lo tanto se trata de entornos creados por y para humanos. En este tipo de entornos para estos autores,

los conceptos como estancias, objetos y propiedades como el tamaño y forma de las estancias, adquieren importancia en las tareas de representar el conocimiento y generar eficientes comportamientos en el robot.

El paradigma de desarrollo de mapas semánticos utilizado está basado en propiedades del espacio. Estas propiedades pueden ser vistas como atributos que caracterizan las entidades de espacio discreto identificadas por el robot, tales como lugares, estancias, o ubicaciones. Adicionalmente, las propiedades las hacen corresponder con conceptos humanos y aportan otra capa de semántica espacial compartida entre el usuario y el robot. Cada propiedad está conectada a un modelo de información sensorial. Los conceptos de alto nivel, como tipos de estancia, son definidos mediante las propiedades. El problema del mapeado semántico queda definido como la estimación de la unión de la distribución de probabilidad de las etiquetas que categorizan una estancia y todos los valores de las propiedades de todos los lugares. Los tipos de propiedades que contemplan son:

- Objetos. Cada clase de objeto se corresponde a una propiedad asociada al lugar. Una ubicación determinada tiene prevista que aparezca cierto número de un tipo determinado de objeto, y se observan una cierta cantidad de ellos.
- Puerta. Determina si una ubicación está determinada en una puerta.
- Forma. La forma geométrica de la estancia extraída de la información de sensores láser.
- Tamaño. El tamaño relativo de la estancia extraído de la información sensorial del láser.
- Apariencia. Apariencia visual de un lugar.
- Espacio asociado. La cantidad de espacio libre visible alrededor de un marcador de posición no asignado a ningún lugar.

El mapa conceptual queda representado en la figura 2.2. Cada instancia concreta de estancia es representada por un conjunto de variables aleatorias, una por cada propiedad asociada a ese lugar.

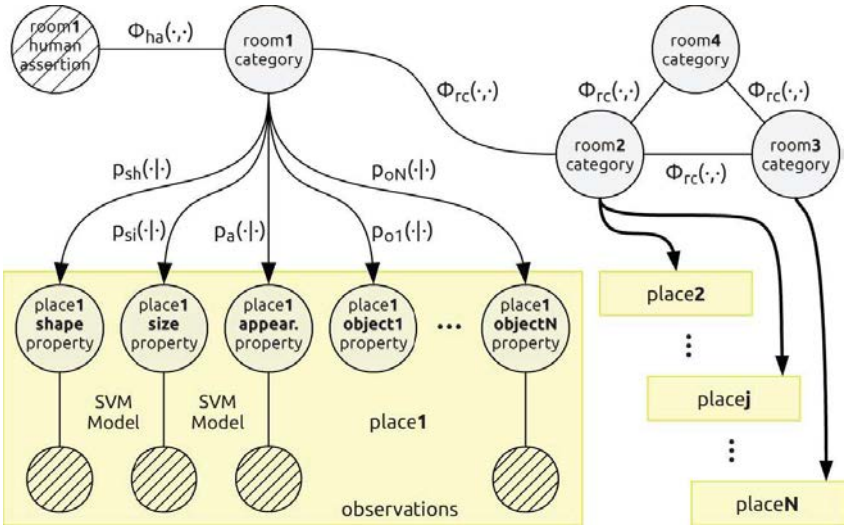


Figura 2.2: Estructura del modelo de grafo del mapa conceptual de [101]

Siendo conscientes de las implicaciones de la elección de la tecnología y del nivel de abstracción a emplear en el mapeado, en [152] consideran de gran importancia la aplicación que tiene el mapeado en robots de servicios y presentan un modelo híbrido de mapa semántico. Clasifican los mapas de navegación en dos categorías: los mapas métricos y los mapas topológicos. Para llegar a un mapeado con las ventajas de ambas categorías y reduciendo las limitaciones, destacan los mapas híbridos [13]. Todos pertenecen a los mapas tradicionales y se centran en representar la estructura geométrica del espacio, describiendo las coordenadas cuantitativas y la conectividad entre localizaciones. Sin embargo, las funcionalidad de las ubicaciones y la complejidad del espacio parcial no es considerado. Tampoco se usan los conceptos semánticos de alto nivel para interactuar con las personas. Esta situación donde el planificador del robot, la localización y la navegación basada en este tipo de mapas, no satisfacen las necesidades de las tareas de servicio del robot, preocupan a los autores de [152] y proponen un mapa semántico, como mejor opción frente a un mapa cognitivo.

Los mapas cognitivos se inspiran de la manera en la que los humanos representan el espacio. La ciencia cognitiva se combina con la inteligencia del robot, con el resultado de que se imita el comportamiento inteligente y la actividad mental de los mecanismos de atención selectiva de los humanos, se focaliza en la experiencia y entendimiento del entorno y se construye la habilidad humana mediante la simulación

de conocimiento por un mapa cognitivo. Varios autores han trabajado en psicología cognitiva e inteligencia artificial. Hace tiempo que Kuipers [72] sugirío la existencia de cinco clases diferentes de información (topológica, métrica, rutas, características fusionadas y observaciones).

En el comportamiento del aprendizaje en el proceso de construcción de mapas cognitivos, se extrae conocimiento de la información actual (percepciones en el momento) y la información histórica (el equivalente al recuerdo o conocimiento anterior). Dentro de los intrincados problemas contra los que hay que enfrentarse en la construcción de mapas cognitivos, destacan la implementación de operaciones cognitivas complejas basadas en la tarea presente y en un comportamiento flexible y la imitación de los mecanismos cerebrales humanos [143].

Actualmente sigue siendo un problema vigente en el mundo de la robótica. En [120] presentan un sistema de navegación en el robot móvil de la figura 2.3 que emplea un mapa cognitivo. La construcción de este mapa se realiza con un algoritmo RatSLAM basado en RGB-D propuesto en [136]. El RatSLAM [82] es un modelo computacional de hipocampo desarrollado con CAN de tres dimensiones que traducen la información de la posición del robot (x,y,θ) en posición de celdas. Las CAN (Continous Attractor Networks) se emplean a menudo para modelar las conductas de lugar, dirección del frente, y rejillas de celdas, especialmente en temas robóticos [127]. Una CAN es un tipo de red neuronal con un array de unidades neuronales con conexiones dependientes de un peso. El sistema llamado RatSlam recibe el nombre debido a que los autores consideran que apunta al entendimiento actual de codificación espacial en el cerebro de las ratas para el desempeño del aprendizaje y recuperación en tiempo real de mapas semimétricos en robots reales. Las *células de posición* forman parte del núcleo de este sistema. Representan los tres grados de libertad de la posición de un robot usando una versión tridimensional de CAN. La actividad envolvente de la estructura de celdas de posición, junto con el gran tamaño del entorno, resulta en que células de posición individuales apuntan a múltiples localizaciones distintas en el entorno. A cada localización donde las células apuntan, el nivel de activación se incrementa mientras el vehículo se mueve a través de donde apunta la celda, y se decrementa cuando el vehículo lo abandona. La actividad de apuntar de una celda de posición es más alta cuando una rata está localizada en una ubicación particular. Un grupo de

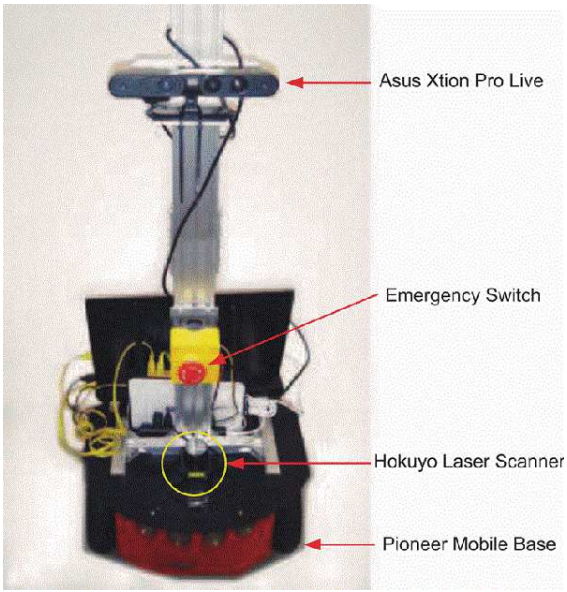


Figura 2.3: Robot móvil NECO-II

unidades altamente activas forman un paquete de actividad. La topología del mapa cognitivo se genera por el desplazamiento del paquete de actividad respecto a la información odométrica. Además, las células de posición están asociadas con pistas visuales para representar lo que está viendo el robot. Cuando varias escenas familiares están viéndose continuamente, el mapa es corregido.

En [120] mejoran el resultado del mapeado con el uso de imágenes RGB-D, aportando información de profundidad invariable ante las condiciones de iluminación. El trabajo desarrollado en esta tesis también emplea imágenes con información de profundidad, con el mismo sensor. Un punto crítico en sus experimentos, a lo que dan gran importancia, es a la tasa de aciertos en la tarea de recordar información del sistema que emplean, pues es el único criterio para evaluar la deriva de la odometría para realizar una precisa estimación de la posición actual del robot. La entrada RGB-D es considerada como un dato multi-modal. La diferencia de intensidad de color se usa para comparar imágenes RGB mientras que la diferencia en distancias se usa para comparar imágenes de profundidad. Además, tratan de disminuir el tiempo de procesamiento usando sólo una pequeña fracción de experiencias cerca de la posición actual en sus comparaciones.

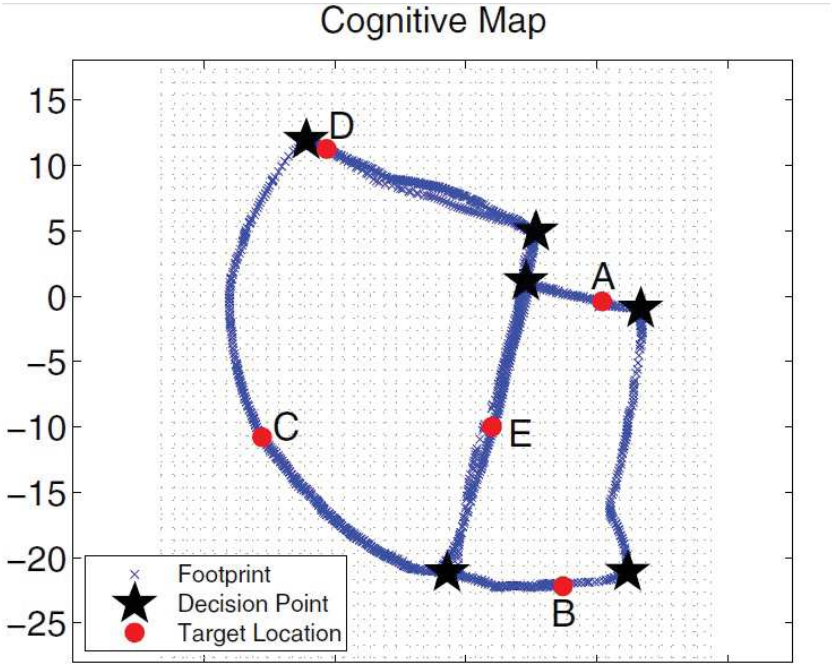


Figura 2.4: Mapa cognitivo generado con una versión del algorito de construcción de mapas cognitivos basado en RGB-D.

El mapa cognitivo generado por su versión del algoritomo de mapeo cognitivo basado en RGB-D se muestra en la figura 2.4. El proceso de mapeo tomó 2351 segundos para viajar 548.2916 metros. se tomaron 3733 imágenes RGB-D. Los puntos circulares (A,B,C,D,E) fueron los objetivos de destino y las marcas de estrella representan los puntos de decisión en los que el planificador extrajo direcciones de movimiento. En [108] dan más detalles de la precisión de este algoritmo, con el que los autores de [120] están satisfechos. A pesar de ello, proponen varias mejoras a su trabajo como especificar un localización concreta dentro de una estancia o implementar el sistema en el marco de un sistema operativo como ROS. Estos temas son tratados en esta tesis.

Se han estudiado otros trabajos que presentan una descripción más formal y genérica, presentando algoritmos para el aprendizaje de mapas cognitivos no necesariamente vinculados a la navegación de robots móviles [61]. Estos autores dirigen sus estudios al área de la ingeniería informática que desea crear mapas de conocimiento.

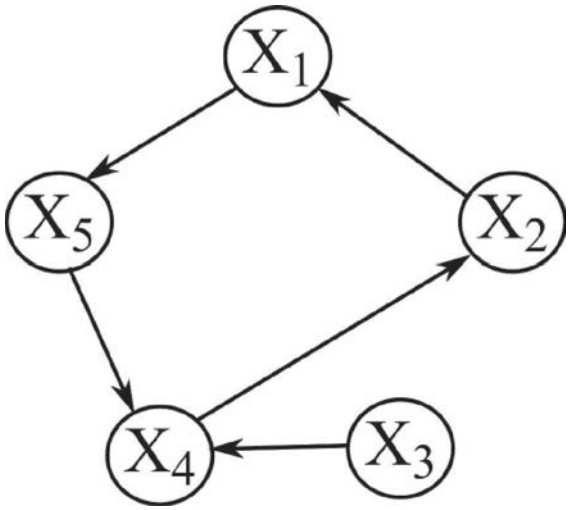


Figura 2.5: Estructura mapa cognitivo.

Es interesante desde el punto de vista del desarrollo de esta tesis enfocada a la navegación semántica el análisis de estas tecnologías de inteligencia artifial, vistas como una posible herramienta aplicable al problema de navegación en robótica. La base de la estructura de un mapa cognitivo es un grafo dirigido en la forma [16]:

$$\langle X, R \rangle \tag{2.1}$$

donde $X = [X_1, ..., X_n]$ - el conjunto de conceptos, n - el número de conceptos; $X_i(t)$ - el valor del concepto i; $R = \{r_{j,i}\}$ - matriz de relaciones, $r_{j,i}$ - la relación entre el concepto j e i.

$$X_i(t+1) = F\left(Xi(t) + \sum_{j=1^n, j\neq i}^{N} r_{j,i} \cdot X_j(t)\right) \tag{2.2}$$

En ese trabajo se usa un modelo dinámico no lineal de un mapa cognitivo descrito por la relación 2.2, donde $F(x)$ - función de estabilización; t - tiempo discreto.

La figura 2.5 presenta un ejemplo de estructura de mapa cognitivo.

Los valores de los conceptos son calculados hasta que un mapa cognitivo alcanza uno de los siguientes estados [25]:

- Un estado estable o inestable

- Un estado caótico

En el trabajo de Jastriebow, destacan la importancia del análisis de estabilidad, especialmente para modelar sistemas dinámicos complejos con abundantes relaciones entre conceptos. En el trabajo citado, presentan la descripción de mapas cognitivos y de algoritmos multi-pasos para el aprendizaje de CM (cognitive maps). La ventaja de aplicar estos algoritmos es que mejoran la funcionalidad del sistema de aprendizaje reduciendo el porcentaje medio de error de predicción, aumentando la velocidad de la convergencia del aprendizaje y obteniendo una mejor estabilización. Entre los experimentos realizados se puede ver, por ejemplo, la estructura del mapa cognitivo inicializado (figura 2.6) resultante del aprendizaje sobre el diagnóstico de hepatitis. La información empleada corresponde a datos reales numéricos y simbólicos tomados del UCI Machine Learning Repository [22]. El aprendizaje del mapa se basó en varios parámetros de 141 registros usando el método. Se emplearon otros 14 registros para los test. Los conceptos de dicho mapa fueron los relevantes en la tarea del diagnóstico. Por ejemplo:

- X_1 - Clase: DEFUNCIÓN, SUPERVIVENCIA.
- X_2 - Edad: 10, 20, 30, 40, 50, 60, 70, 80.
- X_3 - Sexo: Hombre, Mujer.
- X_4 - Antivirales: Sí, No.

Y así hasta obtener un conjunto de veinte conceptos.

También se ha considerado interesante estudiar las técnicas de construcción de mapas cognitivos aplicados al reconocimiento de escenas. Aunque el objetivo de esta tesis se centra en aplicaciones de navegación en entornos humanos, las soluciones desarrolladas para exteriores pueden ser de utilidad e inspiración. En [109] aportan un método para construir una memoria visual que permite reconocimiento espacial sin la necesidad de guardar información visual. Por un lado extraen información visual y por el otro, la construcción de la memoria visual se realiza usando la arquitectura Fuzzy Art [21].

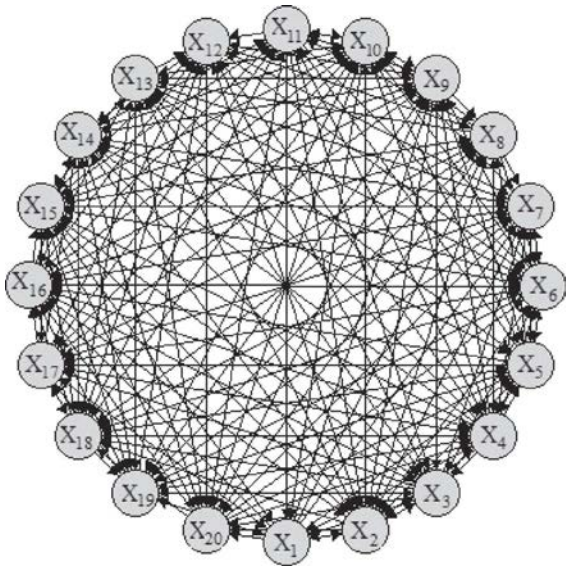


Figura 2.6: Estructura mapa cognitivo II.

La idea la defienden como una imitación del proceso natural de conocimiento espacial en animales y humanos, emulando cómo los animales reconocen un lugar visitado y construyen un mapa cognitivo. Su propósito es construir un mapa coginitivo usando células de visión para robots móviles, empleadas para la memorización del espacio. El reconocimiento de escena es permitido por una Memoria Visual compuesta de estas células de visión. Implementan mecanismos de asociación de datos que describen el entorno del robot para reconocer lugares ya visitados [139]. En los métodos de categorización de características (feature matching), los datos de los sensores son normalizados por la extracción de información útil y entonces se correlaciona con todas las plantillas de la base de datos.

En [109] usan una estructura diferente, orientada a poder reconocer vistas evitando los pasos de guardar la información visual extraída y el proceso de correlación (matching). Su sistema consiste en construir una Memoria Visual incremental usando Fuzzy ART y las características visuales son usadas como señales de entrada de la Memoria Visual para recordar un lugar visitado o crear una Célula Visual representando la escena percibida (ver la figura 2.7). Además de esto, cuentan con un proceso bio-inspirado de Atención Visual (VATT), consistente en procesar cierta parte de la escena con más énfasis, mientras que el resto es desestimado o suprimido. EL modelo

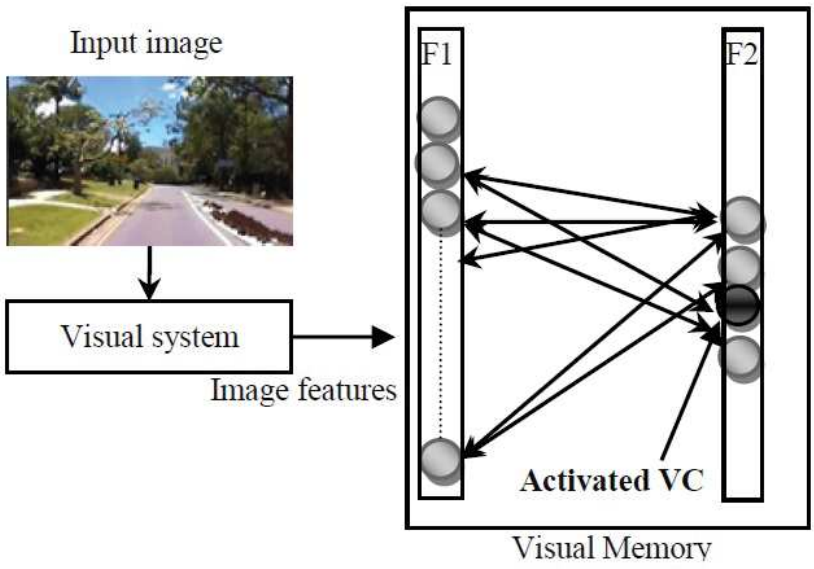


Figura 2.7: Módulos de un mapa cognitivo basado en Memoria Visual.

de VATT debe permitir al robot mirar sólo a una específica región de la imagen. Después, realizan una conversión del espacio de color, para obtener la más simple y rápida característica que proporcioha todas las propiedades invariantes requeridas. Una característica ampliamente usada para reconocimiento de escenas es el histograma de color, que describe la distribución de color de una imagen. Por el hecho de que el robot no necesariamente estará en el mismo punto cuando vuelve a visitar el lugar, tienen en cuenta 4 regiones de la ventana del VATT. Tras el proceso de aprendizaje, cada célula visual responde cuando se mira hacia la misma escena. Dejan pendientes como trabajos futuros investigar en la inclusión de un aspecto métrico en el modelo cognitivo del espacio, pues consideran insuficiente depender únicamente de información visual. Los autores también indican que esta idea está explotada en la construcción de mapas cognitivos y reconocimiento de lugares [121], además de ser aplicada en navegación de robots móviles. En estos puntos es donde tiene sentido estudiarlo para esta tesis. En [146], una Memoria Visual dinámica es usada para almacenar la información adquirida de una cámara en continuo movimiento y un sistema de atención que elige hacia dónde mirar. Con esto componen una Memoria Visual 3D a partir del detector de objetos (para identificar formas básicas y caras humanas que estuvieran

presentes en la imagen actual) y del mecanismo de predición que permite predecir con antelación los items almacenados.

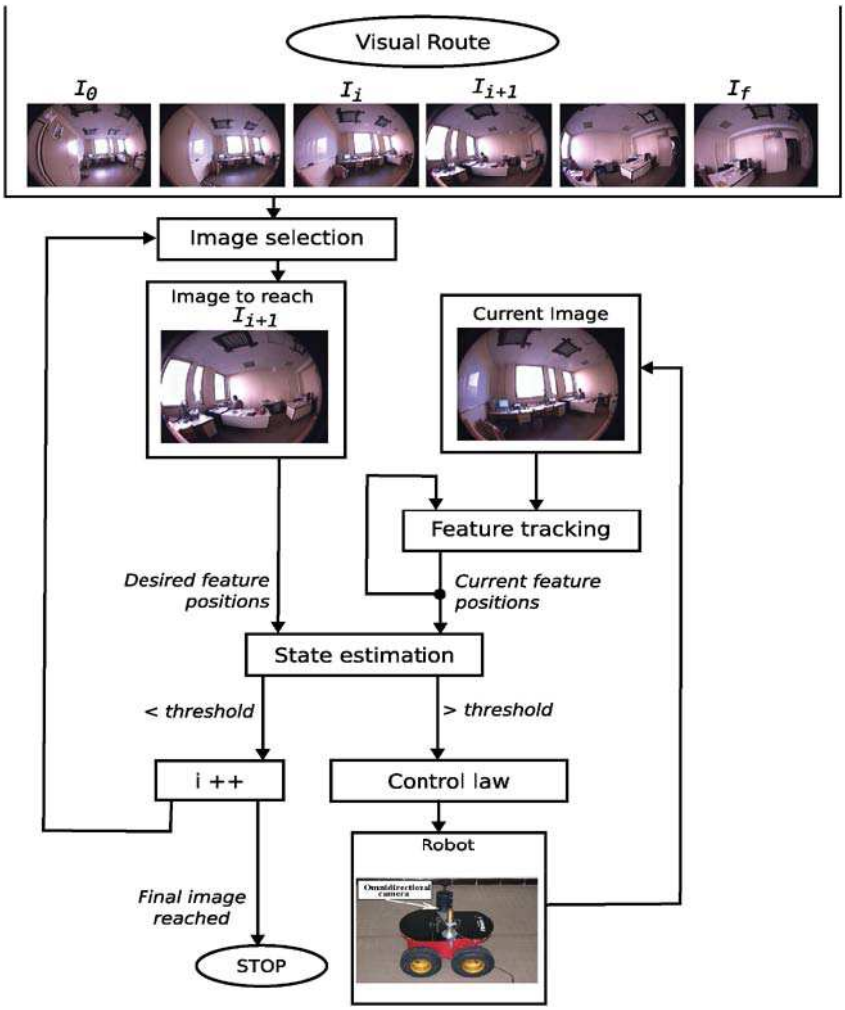


Figura 2.8: Principio del paso de navegación on-line. El robot se controla empleando la imagen actual y la imagen deseada de la ruta visual extraída de la memoria visual.

En [29], proponen un aprendizaje del entorno realizado mediante un grafo de caminos visuales organizados topológicamente y proporcionando una Memoria Visual del entorno. Se basan en la conducta natural de razonamiento humano ante la navegación en un entorno desconocido. La tendencia es memorizar algunos puntos de referencia a lo largo del camino trazado, para emplearlos como checkpoints en futuros trayectos. El sistema de control diseñado no necesita ningún planificador explícito

(path-planning) off-line. La navegación la realizan en un paso on-line, resumido en la figura 2.8. El usuario guía al robot a lo largo de varios caminos en cada estancia y se almacena e indexa un camino visual ${}^{k}\Psi_i$ como el camino i-ésimo en la estancia k-ésima. Un camino visual ${}^{k}\Psi_i$ es un grafo dirigido con pesos compuesto de n imágenes clave sucesivas (vértices).

$$ {}^{k}\Psi_i = \{r\Gamma_p^i | i = \{1, 2, ..., n\}\} \tag{2.3} $$

Además, plantean dos hipótesis, asumiendo que el movimiento del robot sería como el de un automóvil que va únicamente hacia adelante y el control se basa en visión. Formalmente describen las dos hipótesis como sigue:

Dadas dos situaciones ${}^{R}\mathcal{F}_i$ y ${}^{R}\mathcal{F}_{i+1}$, respectivamente asociadas al robot móvil cuando dos sucesivas imágenes clave $\mathcal{I}_i$ y $\mathcal{I}_{i+1}$ de un camino visual Ψ adquirido, existe un camino admisible Ψ de ${}^{R}\mathcal{F}_i$ a ${}^{R}\mathcal{F}_{i+1}$ para un vehículo tipo automóvil cuyo radio de giro es limitado y únicamente mueve hacia adelante.

Dos imágenes clave sucesivas $\mathcal{I}_i$ y $\mathcal{I}_{i+1}$ contienen un conjunto $\mathcal{P}_i$ de características visuales emparejadas, que puede ser seguido a lo largo de un camino representado entre ${}^{R}\mathcal{F}_i$ y ${}^{R}\mathcal{F}_{i+1}$ y que permite la operación de la ley de control.

En la figura 2.9 se puede ver cómo construyen la memoria visual. Una cuestión a tener en cuenta en esta forma de implemetación es que el destino del robot también se da como una imagen visual de uno de los caminos visuales. Sin embargo es una arquitectura tenida en cuenta para combinar con una navegación semántica.

Entrando en conceptos de creación de mapas semánticos (sin ser lo mismo que mapa cognitivo), se encuentran trabajos que exploran este concepto y su aplicación en robótica móvil. En [92] dejan claro que el conocimiento semántico puede ayudar a un robot en su tarea de alcanzar un destino. Y parte de este conocimiento tiene que ser debido a objetos, utilidades, eventos o relaciones en el entorno del robot. La estructura de datos que soporte la información relativa al espacio sobre este entorno es el *mapa*. Un mapa semántico incrementa los mapas típicos geométricos y/o topológicos con información sobre entidades (objetos, funcionalidades, etc) que están localizados en el espacio. Esto implica la necesidad de añadir algún mecanismo de razonamiento con cierto conocimiento previo. De esta manera, llegan a una definición de mapa semántico:

Figura 2.9: Construcción de la memoria visual: En las estancias (a) y (b) y el pasillo (c), el camino ${}^{r}\Psi_{p}$ fue aprendido teleoperando el robot.

Un mapa semántico en un robot móvil es un mapa que contiene, además de información espacial sobre el entorno, asignaciones de características mapeadas de entidades de clases conocidas. Además del conocimiento de esas entidades, independientemente del contenido del mapa, debe estar disponible algún tipo de base de conocimiento con un asociado motor de razonamiento para inferencias.

En el caso del trabajo sobre el que se está hablando, los autores se centraron en diferenciar los siguientes elementos del entorno: Pared, Puerta, Suelo y Techo. Crearon reglas en prolog para implementar las restricciones de las características que diferenciaban las clases, siendo estas puramente visuales. En la figura 2.10 se puede ver la red de restricciones que definían estos elementos, mientras que la figura 2.11 se ven pruebas de detección de objetos en el que incluyen el extintor, una impresora y un árbol de interior en una maceta. El mapeado de bajo nivel era un SLAM 3D.

Una vez renderizada la imagen, realizaban una clasificación teniendo en cuenta el contorno de los objetos aparte de las imágenes de profundidad y de cámara. Después,

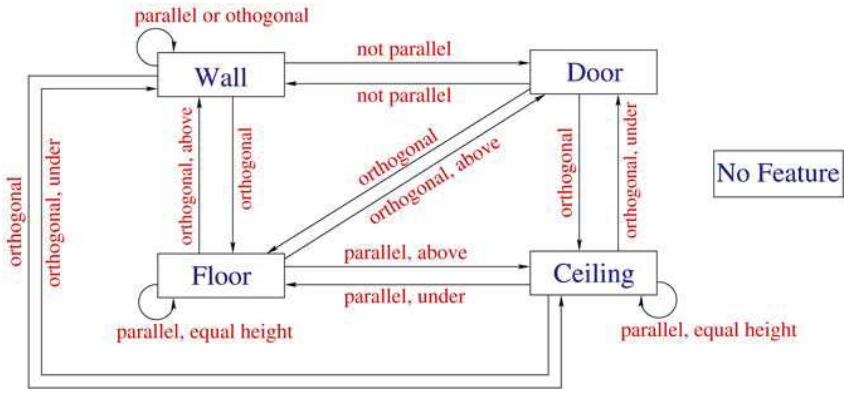


Figura 2.10: Red de restricciones para una interpretación burda del una escena.

seleccionaban los puntos correspondientes a la proyección 2D del objeto (ray tracing). Luego se hacía un emparejamiento con un modelo 3D en esos puntos, seguido de un paso de evaluación.

En [90] también implementan un mapa semántico, atendiendo a la necesidad de clasificar regiones del espacio. Presentan métodos para reconocer y clasificar automáticamente espacios en regiones semánticas separadas y usar esa información para generar un mapa topológico de un entorno. La construcción del mapa semántico se realiza con un guía humano. La partición de este mapa proporciona una clasificación probabilística del mapa métrico, asignando etiquetas, dejando para un trabajo futuro la asignación automática de estas etiquetas (un tema posteriormente solucionado en otros trabajos y en esta misma tesis). El tratamiento simbólico del mapa y un camino tomado por el planificador se pueden ver en la figura 2.12.

Para estos autores, la literatura del mapeo semántico se ha enfocado en desarrollar técnicas de mapeo que puedan funcionar soportando interacciones con humanos, de modo que compartan la representación del espacio. Una de las estrategias que se usan es representar la relación entre un lugar y el conocimiento asociado a él (incluyendo la funcionalidad del lugar y la localización de los objetivos). A diferencia de trabajos como [87], que construyen un mapa topológico sobre un mapa geométrico, proporcionan una clasificación continua del mapa geométrico en regiones etiquetadas semánticamente. Esta digribución multivariable se modela como un modelo Gaussiano. Cada región espacial es representada usando una o más Gaussianas en su marco de coodenadas

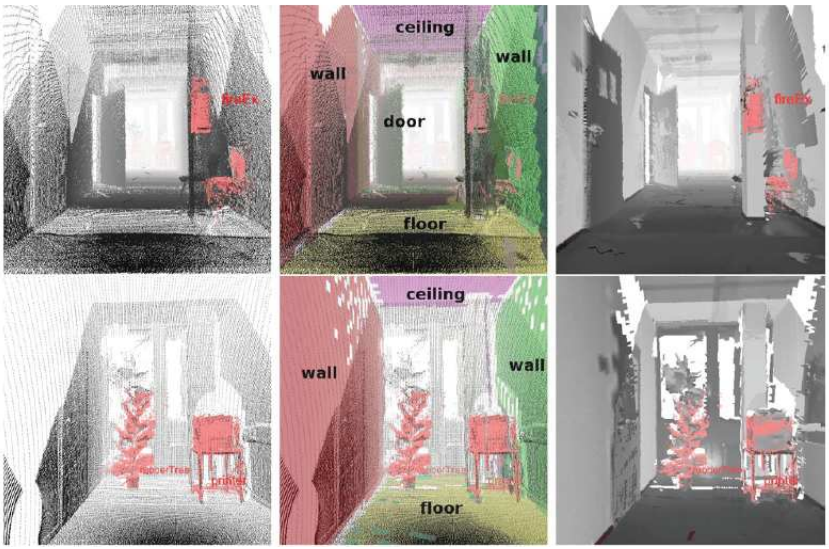


Figura 2.11: Renderizaciones del un mapa semántico 3D en una oficina.

del mapa geométrico. Así pues, una región con etiqueta L y n Gaussianas, cada una de ellas con significado μ y covarianza Σ, es representada como sigue:

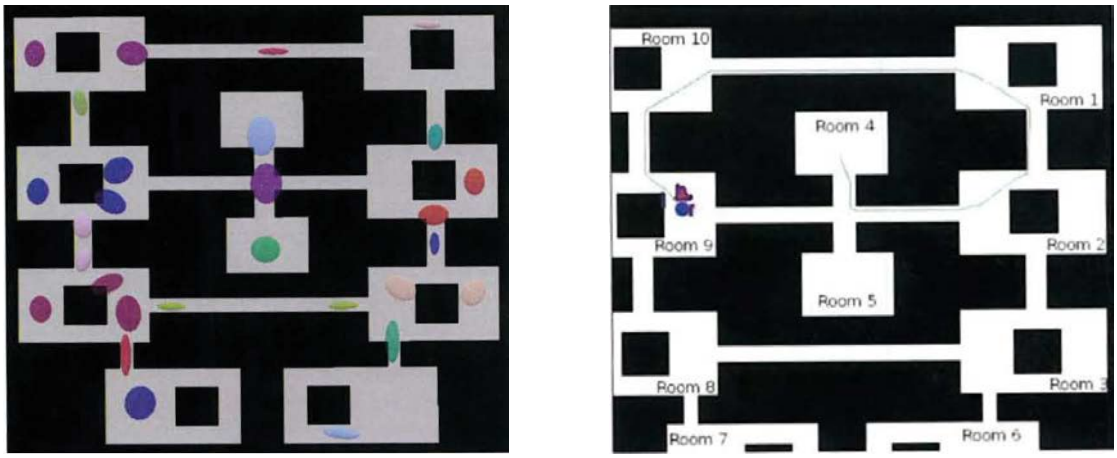
$$Region = \{L, \{\{\mu_1, \Sigma_1\}, \{\mu_2, \Sigma_2\}, ..., \{\mu_n, \Sigma_n\}\}\}$$

En ese trabajo, por lo tanto, representan el mapa semántico como una colección de regiones. Un mapa con m regiones, se representaría como sigue:

$$Map = \{region_1, region_2, ..., region_m\}$$

Después, evaluan la distancia de Mahalanobis de la posición actual del robot x más cercana a las etiquetas codificadas como regiones gaussianas (ecuación 2.4, y eligen la región más cercana según esta métrica, con lo que el robot puede preguntar por su creencia del nombre de la región que esté actualmente ocupada usando esta partición de mapa semántico. Permite clasificaciones probabilísticas de las regiones del mapa.

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)} \tag{2.4}$$



(a) Visualización de regiones Gaussianas. (b) Camino del robot.

Figura 2.12: Una visualización de las regiones Gaussianas representando las estancias y pasillos, junto con el camino del robot replanificado para llegar desde la estancia 9 hacia la estancia 4.

Kuipers propuso un modelo cuantitativo y cualitativo de conocmiento de espacio a gran escala [71], basado en múltiples representaciones de interacción y sirve de base para los anteriores autores en la representación de relaciones de objetos, acciones y dependencias del entorno. A menor escala de espacio, se situaría el trabajo de Beeson et al. [11], proporcionando una representación del entorno de trabajo más específica.

2.2.2. Robótica cognitiva con conocimiento del entorno

Se han estudiado trabajos de otros autores que llevan a un robot con capacidad cognitiva a realizar tareas en un entorno del cual ha adquirido cierto conocimiento. Un caso interesante y que lleva desarrollándose varios años es el proyecto RoboHow. Este proyecto es una iniciativa europea fundada en el año 2012, afronta el desafío de crear sistemas que puedan ayudar a robots a aprender y compartir información entre ellos, imitando el proceso de aprendizaje humano. Sobre este trabajo hay publicaciones como [12] donde prueban las habilidades desarrolladas en un robot PR2 para realizar tareas culinarias (Figura 2.13). En dicho trabajo, describen un experimento que consiste en la preparación de una pizza. Esto implica que el robot combine varias habilidades complejas. El robot tiene que reconocer objetos tales como el bote de

Figura 2.13: Robot PR2 preparado para cocinar.

ketchup en el frigorífico y la cuchara en el cajón. Además, debe monitorizar cómo la pizza se deforma mientras se le pasa el rodillo. Luego el robot vierte la salsa de tomate sobre la masa mientras vuelve a monitorizar el proceso, asegurándose de que la superficie queda cubierta. El robot debe poder ofrecer una manipulación móvil autónoma: tiene que navegar hacia el frigorífico, abrir la puerta y coger el bote de ketchup. El robot también debe **razonar** sobre cómo realizar las acciones. Por ejemplo, tiene que decidir si utiliza una o dos manos para coger los objetos, dónde agarrarlos y cómo soltarlos.

En [12] explican cómo emplean el servicio de conocimiento basado en la nube llamado OPENEASE para realizar sus avances en robótica. Para las aplicaciones web que hacen de interfaz con el software del robot emplean herramientas propuestas en trabajos como [2]. Respecto al contexto de sistemas de procesamiento de conocimiento basado en web, en [150] se propone un sistema llamado ClioPatria. En dicho sistema, los usuarios pueden enviar consultas a una base estática de conocimiento. En vez de permitir a los usuarios escribir consultas en Prolog, ellos usan un lenguaje similar a SQL denominado SPARQL, que es internamente convertido en consultas Prolog. En [114] se presenta un motor de razonamiento llamado RoboBrain, el cual incorpora múltiples modalidades de datos incluyendo símbolos, lenguaje natural, sensores táctiles, trayectorias del robot y características visuales donde el conocimiento se adquiere

de fuentes como la interacción física, bases de conocimiento en web y representaciones aprendidas. En [50] se revisan las maneras en las que la computación en la nube aplicada a la robótica tiene potencial para mejorar la representación.

2.3. Adquisición de conocimiento del entorno

La adquisición de conocimiento del entorno es una habilidad crítica para la interacción de un robot con su entorno. Hay mucha información que se puede tener en cuenta. Según [101] se consideran únicos por contemplar todas las fuentes de información que han sido vistas hasta el momento. Combinan la información sobre la existencia de objetos en una estancia, marcas o señales, la estructura topológica, la apariencia y tamaño de la estancia, la geometría del espacio y los datos proporcionados por una interacción con usuarios humanos. Todo ello está integrado con un conocimiento conceptual previo basado en sentido común. Mantienen distintas capas de abstracción, la capa conceptual contiene la representación relacional. Las relaciones pueden estar predefinidas, haber sido adquiridas o haberse inferido. Se consigue así, con dichas relaciones, una estimación probabilística del tipo de estancia donde se encuentra el robot. La ontología del entorno está codificada manualmente basándose en [44] y [157], y se apoyan en una amplia literatura sobre localización, mapeo y clasificación de lugares como se explica en [34], [35], [138] [153], [107] o [104]. Una idea interesante que también tienen en cuenta es la conectividad entre habitaciones por probabilidad, lo que permite razonar sobre espacio desconocido [154]. La cocina está frecuentemente conectada con un pasillo, pero no a un baño, por ejemplo. Esta idea se ve también en [8] donde emplean probabilidades sobre lo que puede haber tras una puerta. En un paso más allá, realizan predicciones sobre terreno inexplorado soportando todas la hipótesis sobre lo que podría ser una región inexplorada.

En [47] presentan un sistema construido a partir de otro trabajo [106] donde describen un procedimiento de adquisición de conocimiento basado en interacción multimodal con el usuario y construyen un mapa semántico. El mapa se basa en dos partes, la primera se obtiene con técnicas de SLAM (*Simultaneous Localization And Mapping*), que proporciona la información métrica del entorno. La segunda parte la forma una interfaz multimodal que permite al usuario señalar los elementos del

entorno y asignarles un rol semántico. La novedad que ofrecen con esto se encuentra en dos puntos:

- La representación del entorno se extrae automáticamente del mapa métrico y se vinculan etiquetas mediante la interacción con el usuario.
- El proceso de construcción y actualización de la representación es un proceso continuo e interactivo.

Esta tesis afronta la adquisición de conocimiento del entorno con ventajas similares a las comentadas aunque no se limita a las mismas, puesto que la interacción con el usuario recoge conceptos más amplios que la correlación entre entidades físicas y conceptuales. Además, la adquisición del conocimiento del entorno también viene dada por otros medios aparte de la interacción, con lo que supone un avance respecto al estado del arte.

Según algunos autores como Kuipers [71] o Gemignani et al. [47], los sistemas de mapeo semántico se pueden agrupar en dos categorías principales: las que emplean métodos completamente automáticos para la adquisición del conocimiento necesario para el mapeo semántico, y las que involucran a un usuario humano para esta tarea.

2.3.1. Adquisición enteramente autónoma

En esta categoría de métodos completamente autónomos no se considera la interacción humana. En [47] dividen esta categoría en tres conjuntos diferentes:

- El primer conjunto es el de los métodos que adquieren características del entorno empleando las medidas métricas tomadas con los sensores láser. Con estos datos extraen información de alto nivel y realizan el etiquetado. Un ejemplo de esto se puede encontrar en [44], donde el conocimiento del entorno es representado como un mapa topológico aumentado con conocimiento semántico mediante una vinculación denominada *anchoring*.
- El segundo conjunto de técnicas usan una clasificación y agrupamiento (*clustering*) para segmentación automática y etiquetado de mapas métricos. En esta categoría entran trabajos como [49] (empleando AdaBoost), [20] (empleando

agrupamiento espectral) y [42] (empleando campos aleatorios de Voronoi) donde generan mapas topológicos bidimensionales.

- El tercer conjunto de métodos es el caracterizado por emplear técnicas de reconocimiento de objetos y etiquetado de lugares usando características visuales. El trabajo presentado en [153] es un ejemplo de ello. Dentro de la misma categoría está el trabajo descrito en [136], donde combinan información visual con datos de distancias proporcionado por una cámara RGBD.

Uno de los aportes de esta tesis respecto al estado del arte es ampliar este conjunto de categorías añadiendo una cuarta, caracterizada por emplear técnicas de clasificación de lugares basadas en identificar las acciones que realizan las personas que se encuentran en el entorno.

2.3.2. Adquisición asistida por un usuario humano

En esta categoría de sistemas para mapeo aumentado por humanos, el papel del usuario es explotado para asentar símbolos como objetos que son reconocidos autónomamente por robots. En este caso, la interacción humano-robot es normalmente unimodal y típicamente establecida mediante diálogos de voz. En esta tesis la interacción se consigue mediante interfaces de voz o de teclado. En [157] se describe un sistema para crear representaciones conceptuales de entornos de interior. El robot posee un conocimiento previo sobre conceptos espaciales y el papel del usuario consiste en asistir al robot en el proceso de etiquetar los lugares. Algo similar se describe en [90], donde un usuario hace de guía para asistir al robot en el proceso de asociar una región espacial con una etiqueta semántica. En [101] se presenta un algoritmo de mapeado semántico multi-capa que combina información sobre la existencia de objetos y las propiedades semánticas del espacio (tamaño de la habitación, apariencia, etc). Con esa información el sistema realiza una clasificación de estancias pero si se añaden datos proporcionados por el usuario, se integran como propiedades adicionales sobre los objetos existentes. Existen otras estrategias como la presentada por Peltason et al. [97] orientadas al aprendizaje del robot mediante la interacción con el usuario, sin embargo sólo se consideran las estancias y no se permite incluir objetos en el mapa semántico.

En la línea de incluir interacción con lenguaje natural, se encuentran trabajos como [148] donde se permite que el robot aprenda modelos de entornos humanos a partir de descripciones de lenguaje natural. Sin embargo, otros autores suelen preferir la interacción multi-modal para abarcar distintos tipos de información. Kruijff et al. [70] introducen un sistema para mejorar el mapeado con diálogos de clarificación entre humano y robot, usando lenguaje natural. Algo parecido se realiza en [54] donde la información espacial y semántica son asociadas mediante un *etiquetado de eventos* durante un recorrido narrado por parte del usuario que acompaña al robot.

2.4. Clasificación de lugares

En las últimas décadas, hay gran número de investigadores enfocados en la navegación cognitiva, un área que combina la tarea de desplazamiento del robot con un alto nivel de percepción del entorno. Estos navegadores requieren poder etiquetar de alguna manera las estancias o áreas por las que se puede mover el robot. El etiquetado consiste en nombrar dicha estancia o región para identificarla unívocamente respecto a las demás regiones y además asociarla a una categoría que aporta ciertas propiedades, que serán tenidas en cuenta a la hora de realizar la navegación. La información que ofrece la categoría depende del nivel de abstracción permitido por la capacidad sensorial y la ontología empleada. Esto quiere decir que cuanta más capacidad de percibir elementos del entorno tenga el robot, más nivel de abstracción tendrá el etiquetado. La navegación cognitiva requiere poder clasificar las regiones del entorno. En [68], los autores afrontan el problema de la navegación cognitiva o semántica descomponiéndolo en tareas discretas. En ese trabajo comentan los objetivos del reconocimiento de lugares y la categorización de regiones. Inciden en que el navegador necesita utilizar robustas técnicas de aprendizaje automático para gestionar cualquier cambio dinámico del entorno explorado y afirman que el robot debe ser capaz de clasificar y etiquetar lugares, el sistema no debe limitarse en el reconocimiento de lugares diferentes. Está claro que cualquier navegador debe poder diferenciar, por ejemplo, que la habitación 1 no es la habitación 2, si no sería imposible la localización y no podría hablarse de navegación. Pero el navegador cognitivo además utiliza información relativa a la categoría del lugar. Las categorías que se pueden manejar

dependen de la complejidad del sistema, puede ser que el espacio de categorías se reduzca a {*Habitación, Pasillo, Puerta*} o maneje un espacio de categorías del tipo {*Cocina, Salón, Hall, Dormitorio*}. En cualquier caso, el navegador cognitivo asocia categorías a las regiones del entorno.

La literatura acerca de métodos de etiquetado de lugares para navegación de robots es extensa [37] [99] [27]. Una tendencia es identificar regiones de interés del entorno, tales como suelo, pared y puertas [110]. Pero, aunque esto dota al sistema de cierto conocimiento relativo a la navegación, no categoriza el lugar. Esta tarea es abordada en [118] donde ellos diferencian entre pasillos, despachos, salas de lectura and puertas. Además, los autores sopesan las ventajas y los inconvenientes de emplear distintos sensores para el etiquetado semántico de lugares. Por ejemplo, comentan que [119] y [147] están basados en sensores de visión, mientras que [86] y [123] están basados en datos de distancias mediante medidas de láser. Además, comentan un sistema de etiquetado basado en un sistema multi-sensorial descrito en [103].

Se aprecia que etiquetar lugares es un objetivo muy estudiado, habiendo muchas otras técnicas como las que se describen en [3] aplicando la transformada de Houg para identificar pasillos. En [93] entrenan una red neuronal con información odométrica para detectar la posición del robot.

Hay que considerar que una de las posibles áreas beneficiadas por este trabajo es el campo de la navegación de robots móviles, por lo que esto suscita gran interés en este área. Navegadores topológicos pueden ser construidos a partir de los resultados de los clasificadores obtenidos que etiquetan los nodos del mapa. También navegadores semánticos que empleen el método de este trabajo pueden ser construidos sobre un navegador topológico o geométrico. En [86] usan datos de un sistema que devuelve un plano de 360º para distinguir entre habitación, pasillo, puerta y recibidor. Para lograrlo, emplean únicamente datos geométricos. Cuando se emplean mecanismos de visión para reconocer objetos se puede afinar más en la etiquetación de lugares. En [112] usan características *Haar* para obtener el número de objetos específicos presentes en el entorno.

Ninguno de estos sistemas han incluido técnicas diseñadas para reconocer patrones en las acciones de los individuos, es decir, para etiquetar una estancia en función de lo

que las personas hacen en ellas. El sistema desarrollado en esta tesis abre un camino en esa dirección.

2.4.1. Detectando los objetos

El reconocimiento de objetos es otro de los pilares sobre los que se apoya un sistema de navegación semántica. Aunque no es el tema central de este trabajo, es necesario considerar las posibilidades reales que existen actualmente. Hay métodos empleados en otros trabajos de navegación semántica como SIFT [77] combinado con otros métodos empleados en [41], [40] o [76]. Cabe mención también el desarrollo de un sistema de búsqueda visual activa [8], combinando las pistas semánticas para guiar el proceso de búsqueda de objetos. Parten de un entorno desconocido y proporcionan una estrategia de exploración que tiene en cuenta la tarea de buscar un objeto y el planificador adapta el comportamiento de la búsqueda dependiendo de las condiciones actuales. Es una búsqueda indirecta que se conoce muy anteriormente [46], si se busca el teléfono, se busca primero la mesa donde está el teléfono. Se soluciona por tanto el problema de que los objetos a buscar no están normalmente en el rango de los sensores. Y la manera de solucionarlo vuelve a ser el empleo de información semántica.

La habilidad de detección e identificación de objetos se emplea frecuentemente en la tarea de clasificación de lugares requerida por la navegación semántica. En [85] presentan un sistema de reconocimiento de muebles para cumplir con esta tarea. Se emplea un método de aprendizaje automático de un conjunto de partes de modelos CAD (*Computer Aided Design - Diseño Asistido por Ordenador*) de objetos descargados de internet, con lo que consiguen detectar y localizar instancias de esos tipos de objetos en escenarios representados por nubes de puntos tridimensionales. La navegación semántica, por lo tanto, puede obtener un beneficio directo de estos métodos de clasificación de lugares y que permiten encontrar la ubicación posible de un objeto concreto. En esta aplicación centran el interés en [62] donde aprovechan la información semántica para encontrar objetos. Las técnicas de detección de objetos también han sido usadas para crear mapas semánticos [151].

En [6] se propone un algoritmo que permite a un robot móvil diferenciar entre los objetos de un escenario con la idea de utilizar esa información para navegación semántica. Los requerimientos de la detección de objetos aplicados a este campo son

menores que si se aplican a tareas de manipulación y agarre de objetos [15] en cuanto a la precisión de localización. Sin embargo, se encuentran trabajos donde los objetos y sus distancias se emplean para la clasificación de lugares [145]. En otras ocasiones, se limitan a contabilizar la presencia de objetos en una localización para realizar inferencias [144].

Existen numerosos métodos de reconocimiento de objetos aplicados a la robótica móvil que se han desarrollado en las últimas décadas. Algunos investigadores emplean algoritmos basados en SURF (*Speeded-Up Robust Features*) [10]. En esta línea están trabajos como [6, 117, 149], habiendo también combinaciones de este con otros métodos como uno basado en superpíxeles [36]. Otros métodos se basan en *bag of words* (bolsa de palabras) para la clasificación de objetos [33]. Estos métodos producen buenos resultados en la tarea de la detección de objetos debido a la gran cantidad de descriptores que se extraen en cada imagen.

Hay que tener en cuenta que la fase más trascendente de un sistema de detección de objetos es la segmentación de la imagen [105]. En [56] se emplean técnicas de segmentación basadas en el color, mientras que en trabajos como [57] se proponen métodos que emplean técnicas de segmentación de profundidad para procesar la imagen de color y de profundidad proporcionada por el sensor de visión.

2.4.2. Reconociendo las escenas

En esta tesis se emplean alternativas innovadoras para la clasificación de lugares. Entre estas alternativas se encuentra la clasificación de estancias mediante lo que hacen las personas en ese lugar. Las acciones de las personas se tratan de deducir en función a su movimiento y al sonido ambiente.

Detección de personas

Sin emplearse para la clasificación de lugares, la detección de personas es un problema ampliamente abordado. Un sistema completo puede encontrarse en [79]. Ellos realizan una segmentación de bulto, detección de cabeza y hombros y un refinamiento temporal. No obstante como primera aproximación a la tarea que nos compete, el algoritmo de detección de personas que se emplea en esta tesis está orientado solamente a la detección de piernas como han optado otros autores [1].

Detección de sonido ambiente

Respecto a la utilidad de la información de sonido, los trabajos previos de plataformas robóticas móviles que involucran micrófonos han sido orientados principalmente a la localización de la fuente de sonido [23] y la interacción Humano-Robot con diálogo. En cuanto a la localización por sonido, empleando varios micrófonos distribuidos se puede calcular la posición de una fuente sonora en un entorno determinado, como se muestra en [18, 98, 129]. Este principio ha sido empleado con éxito en aplicaciones como sonar submarino, teleconferencia o asistencia a la audición. Además, puede ser empleado para detectar múltiples fuentes activas y pasivas. Respecto a la interacción Humano-Robot, la importancia de una simbiosis entre humanos y robots lleva a una mejora de las habilidades de percepción en robots. En concreto, las habilidades de escucha están siendo estudiadas para que la interacción sea posible en entornos del mundo real [122, 126, 155]

2.4.3. Clasificadores utilizados

Se aprecia la búsqueda constante de obtener un mecanismo que catalogue e identifique una región del espacio. En [130], tratan de diferenciar zonas en base a la utilidad de los objetos que se encuentran. Dentro de cada habitación se diferencian zonas por objetos agrupados por su función. Emplean un clasificador Naive Bayes como en [38] para inferir la identificación del lugar. Dentro de sus reglas, tienen en cuenta el número de objetos y la distancia entre ellos, para poder agrupar mejor y crear un conjunto que represente una zona. Con su método aumentan la efectividad de clasificación que se ve en [142]. El proceso principal de agrupación de objetos parte de los modelos de conceptos y el conjunto de objetos percibidos. Mediante iteraciones sobre estos objetos, se van buscando los vecinos cercanos, se ordenan y se obtiene una lista separada de conceptos, obteniendo el que mejor se ajusta.

En [66] también se apoyan en métodos probabilísticos basados en Naive Bayes para la clasificación de lugares y la obtención de probabilidades de localización de objetos. Es un hecho que algunos objetos tienden a estar o residir en ciertos tipos de lugares del entorno y no en otros. Las relaciones entre objeto-objeto (por ejemplo, un sofá puede estar cerca de un control remoto y viceversa) y objeto-escenario (un sofá,

un control remoto y un televisor están relacionados con una sala de estar) se pueden aprovechar para predecir la localización de gran variedad de objetos y escenarios. Más recientemente se publicó un artículo [65] donde también emplean una aproximación Bayesiana para el mapeado semántico. En ese artículo, Ko et al. combinan un mapeado semántico, topológico y geométrico del espacio y nodos relativos a objetos. La tarea de localización se emplea con métodos de clasificación de lugares.

Una de las técnicas empleadas en esta tesis para aprender las categorías y, por lo tanto, etiquetar estancias en tiempo real, se basa en SVM (*Support Vector Machine - Máquinas de Soporte Vectorial*). Esto es similar a la metodología usada en trabajos de otros autores como Sousa et al. [123]. Ellos enfrentan el mismo problema de la clasificación semántica del entorno, pero emplean datos de sensores de distancia incorporados en robots móviles. Eligen usar directamente las muestras de entrenamiento como entradas, para que todas las posibles situaciones sean contempladas en el entrenamiento, de este modo consiguen un buen porcentaje de clasificaciones correctas. Los datos brutos son transformados en grupos de características geométricas simples desde las cuales se puede extraer la clasificación de lugares. Después se obtiene un clasificador entre distintos tipos de estancia y pasillos. Los datos con los que se entrena la SVM se basan en el área, perímetro, compactación, excentricidad y circularidad (definida como $\frac{perimeter^2}{area}$) extraída de los entornos.

2.5. Navegación semántica

En [157], apoyados en que los humanos adoptamos parcialmente una representación jerárquica de organización espacial [125] [80] y la categorización de las estructuras espaciales de [19] o [111], diseñan un sistema integrado para crear representaciones conceptuales basado en mapas múltiples de diferentes niveles de abstracción. Se incluye un sensor láser, una cámara y un sistema para interacción por voz. Se proporciona un alto nivel de comunicación humano-robot y de representación conceptual pero no hay ningún conocimiento semántico asociado a los objetos. Esta tesis sí se encarga de este asunto, como muestra la sección 4.1.2. Todos los trabajos mencionados se apoyan en una amplia literatura de localización, mapeado y clasificación de lugares en la que

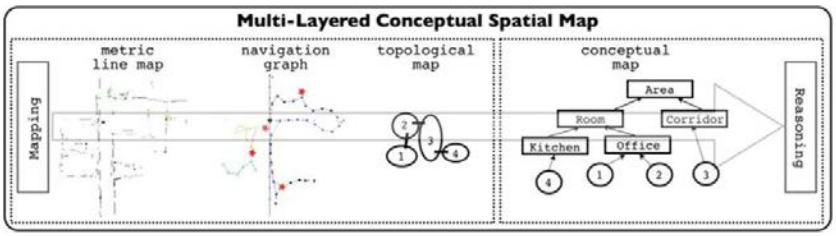


Figura 2.14: Representación de los distintos niveles de mapeo

se incluyen [34], [35], [138] [153], [107] o [104]. Esta representación puede verse en la figura 2.14. En la representación del espacio cuentan con:

- Mapa métrico (SLAM), para la navegación a bajo nivel, pero insuficiente para que un robot pueda entender la información del diálogo sobre el destino.
- Grafo de navegación (Nodos). Aquí la única información que puede extraerse es la diferenciación entre pasillo, habitación y entrada. Se tienen en cuenta las características métricas de la región en la que se encuentra. Los autores consideran esto como un nivel básico de información semántica por llegar a hacer la distinción comentada.
- Mapa conceptual. Por un lado, contiene la ontología innata conceptual, por otro la información extraída de los sensores o la obtenida mediante diálogo. Es la diferenciación clásica de la T-Box (conceptos) y la A-Box (instancias), presente en la literatura de sistemas tradicionales de representación de conocimiento. El empleo de este mecanismo no permite la modificación en tiempo de ejecución de las reglas del conocimiento. El sistema propuesto en esta tesis, al prescindir de reglas propiamente dichas, sí permite la modificación del conocimiento en cualquier momento.

Por ello, es común la división de varios tipos de conocimiento tal como se propone en [157] donde se distinguen los siguientes tipos de conocimiento, en función de la manera de obtenerlo:

- Conocimiento adquirido. Lo que va detectando el robot con sus sensores mientras construye el mapa métrico y topológico (diferenciación entre habitación y pasillo y detección de objetos).

- Conocimiento confirmado. Cuando el robot va acompañado de un usuario que va nombrando áreas y objetos.
- Conocimiento conceptual innato. Referido a lo pre-establecido en la T-Box, las reglas que diferencian los distintos tipos de habitación.
- Conocimiento inferido. Es el conocimiento deducido en base a las reglas y las observaciones. En este caso, la identificación de estancias.

2.5.1. Generalidades de la Navegación Semántica

En la literatura se pueden encontrar publicaciones que describen sistemas de navegación semántica. En esta tesis se ha tenido en cuenta la manera en la que se suelen aproximar al problema para encontrar generalidades entre la distintas aproximaciones y detectar los posibles aspectos mejorables actuales. En [67] afirman que su mapeo semántico abarca:

- Un framework para el mapeo topológico. Donde el entorno explorado es expresado geométricamente y además se genera una abstracción topológica. La necesidad de trabajar con otro navegador con un menor nivel de abstracción es común en los navegadores semánticos. En [87] también incluyen una descripción de cómo generan un mapa topológico, asumiendo que al robot le ha sido dado un mapa del entorno en forma de celdas de ocupación [83]. Siguiendo esta línea, en esta tesis se incluye una interfaz para integrar distintos navegadores de bajo nivel en el navegador semántico.
- Un método de clasificación de lugares. En [67] emplean un método de histogramas basado en la representación de lugares previamente memorizados que presentaron en [68]. Sin embargo, en [87] emplean un método llamado *Boosting*, que es un método general para crear un fuerte y preciso clasificador combinando un conjunto de clasificadores débiles. En concreto, usan el algoritmo *AdaBoost*. Se puede encontrar información detallada del funcionamiento de este algoritmo en [115]. En esta tesis se describen en el capítulo 5 los métodos de clasificación de lugares empleados, destacando un novedoso método basado en lo que hacen las personas en el entorno [32].

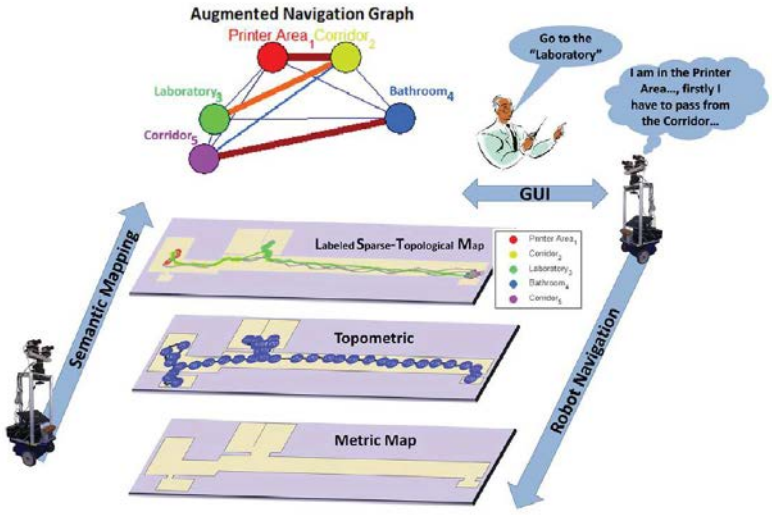


Figura 2.15: Grafo de Navegación Aumentada y la relación entre niveles de mapeado en [67].

- Una partición de lugares no supervisada. En [67] primero segmentan el entorno para tener lugares distinguibles formando un mapa topológico de etiquetado disperso. En esta tesis se resuelve este problema con el mapeado descrito en la sección 4.1 y apoyándose en el diseño de una ontología para distinguir la jerarquía conceptual y en el navegador de bajo nivel para alcanzar la jerarquía espacial.

- Una representación conceptual de lugares detectados. En [67] lo llaman *grafo de navegación aumentada* y se trata de unir los conceptos de la jerarquía conceptual de lugares entre sí. En la Figura 2.15 se muestra el ejemplo de [67] sobre su *grafo de navegación aumentada*, donde clasifican los nodos del nivel topológico en una de las categorías de su jerarquía conceptual. Los navegadores semánticos suelen incluir esta característica pero no se limitan a los conceptos relativos a lugares. En esta tesis, la jerarquía conceptual es ampliada con objetos, utilidades y valoraciones subjetivas. Y es común ver otros navegadores que incluyen una jerarquía conceptual más amplia, como comentan en [101].

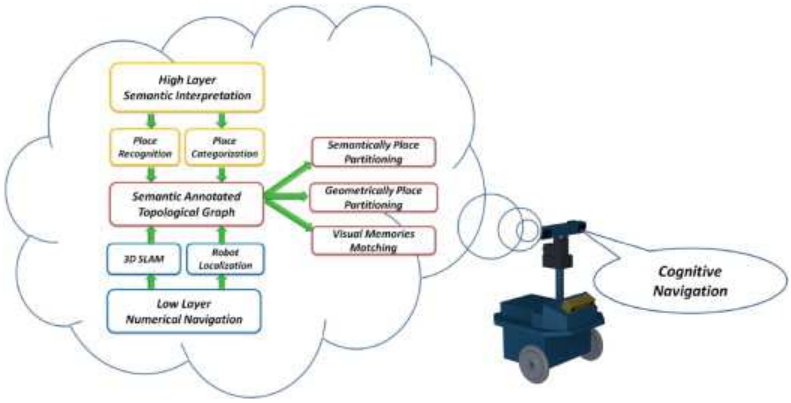


Figura 2.16: Componentes de la navegación semántica en [68].

2.5.2. Componentes de la navegación semántica

En las últimas décadas, la robótica cognitiva aplicada en navegación ha llevado a la combinación de robótica móvil con un alto nivel de percepción. Los navegadores semánticos conciernen a dos conceptos [94]. El primero es el relativo a la capacidad del robot de autolocalización y de generar un mapa métrico del entorno explorado. El segundo es la *interpretación semántica* y se refiere a la capacidad del robot de comprender su entorno. En [68] se comenta que los robots cognitivos deberían poder realizar inferencias semánticas basadas en mecanismos de interpretación del contexto, incluso cuando visitan lugares por primera vez. También reconocen la labor de investigaciones específicas en este área como el presentado en [141], donde experimentan con la clasificación y reconocimiento de lugares mediante una representación probabilística basada en objetos. Es interesante que el trabajo presentado en [68] pretende señalar los objetivos básicos de un *framework* completo de navegación cognitiva. Descomponen el problema en una serie concreta de tareas que constituye su navegador (Figura 2.16) y lo organizan en los siguientes objetivos:

- *La navegación de bajo nivel.* Comprende los atributos numéricos y geométricos que emplea el robot para su localización.
- *La navegación de alto nivel.* Incluye los atributos cognitivos que emplea un robot para realizar inferencias semánticas sobre su localización. Esto lo descomponen en las siguientes sub-tareas:

1. Abstracción espacial. Es la representación de la información espacial aprendida. Engloba los conceptos que en esta tesis se han llamado *conceptuales* pero limitándose a estancias conceptuales como *despacho* o *cocina*.
2. Reconocimiento de lugares. El sistema debe ser capaz de reconocer las instancias de lugares aprendidas para moverse de una localización a otra.
3. Clasificación de lugares. El sistema debe ser capaz de etiquetar y clasificar lugares aunque no disponga de información sobre la categoría de dicho lugar. Es decir, debe poder generalizar y con ello clasificar lugares vistos por primera vez.

- *Una interfaz que comunica el bajo con el alto nivel.*

Estas ideas recogidas en esta sección ayudan a comprender los objetivos presentados en la introducción. Se ha presentado el estado actual de la navegación semántica y de todas las áreas de investigación que se solapan con este propósito, destacando las características comunes con el trabajo realizado en esta tesis y los aspectos donde el presente trabajo puede suponer alguna diferencia.

CAPÍTULO 3

Arquitectura del sistema

"Las grandes ideas son aquellas de las que lo único que nos sorprende es que no se nos hayan ocurrido antes." — Noel Clarasó

En este capítulo se presenta la arquitectura de todas las partes que integran el sistema de navegación semántica. Se ha optado por diseñar una estructura muy modularizada, en previsión de que los distintos módulos puedan ser cambiados para ser implementados mediante otras tecnologías. En el propio desarrollo de esta tesis, ya se probaron distintas implementaciones de algunos módulos para someter a comparación los resultados. Esto es obviamente muy deseable en el mundo de la investigación, por lo que se ha realizado el esfuerzo necesario para que todos los módulos estuvieran correctamente desacoplados entre sí. De este modo se tiene un sistema de navegación que sirve también como herramienta para comparar la eficiencia de las distintas maneras de afrontar cualquier tarea requerida en la navegación semántica.

3.1. Descripción general

En esta sección se introducen la visión global de todos los elementos que forman el sistema de navegación. Hay una fuerte influencia de otros trabajos como [9], [39]

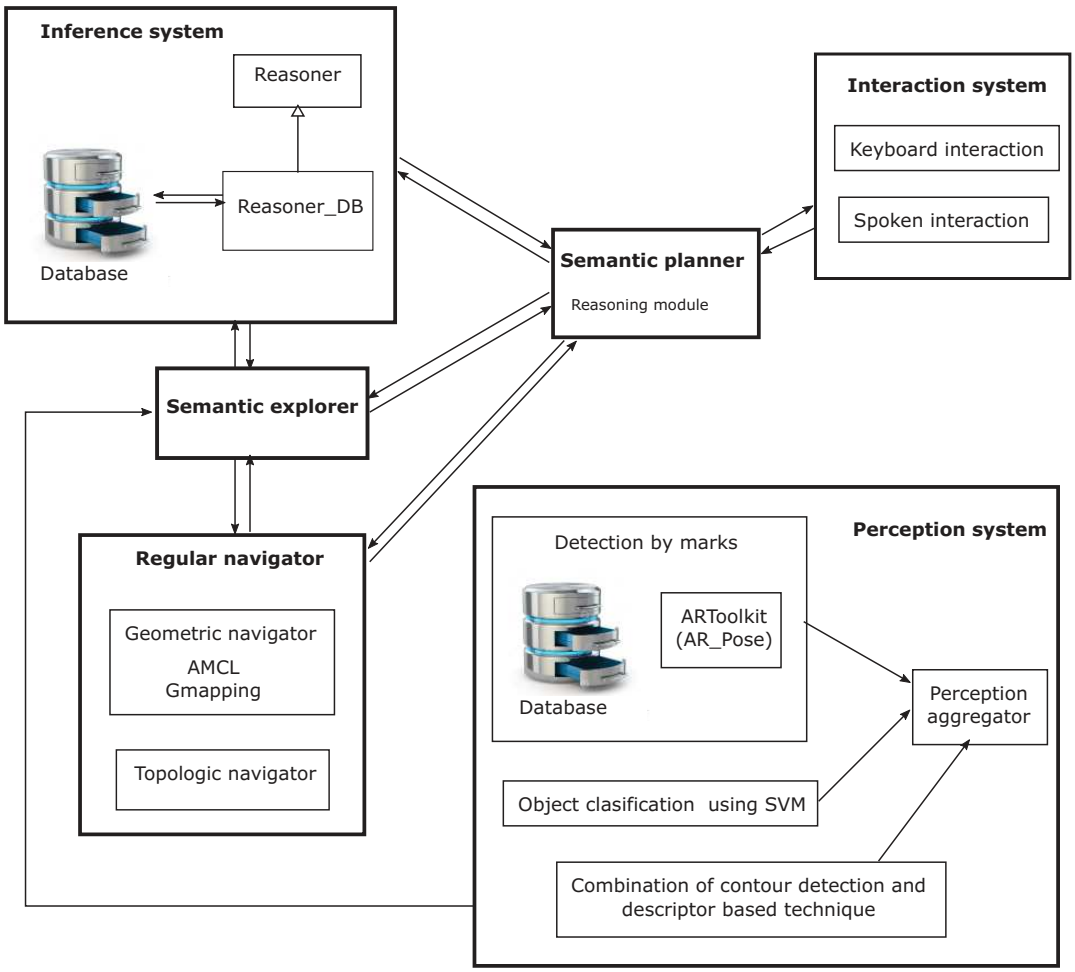


Figura 3.1: Sistema completo de navegación semántica.

en cuanto a las tareas que debe cumplir un robot que se considere autónomo a la hora de navegar, y de los elementos que se necesitan (explorador, planificador, etc). Los módulos principales que requiere el sistema se muestran en la figura 3.1. Casi todos ellos van a ser expuestos en las siguientes secciones de este capítulo. El sistema completo de navegación semántica consta de los siguientes elementos:

- **Sistema de inferencia**. Este elemento es la pieza clave de la navegación semántica, absolutamente imprescindible puesto que es donde se realizan inferencias, se hace deducciones, se extraen conclusiones y por lo tanto, es el

elemento que "razona". Es utilizado tanto por el planificador semántico como por el explorador, que son los elementos que requieren de la capacidad de realizar inferencias. Aunque, en realidad, su presencia en el sistema no se manifiesta como un nodo independiente, sino que es una clase instanciada por los elementos que la utilizan. Su funcionamiento y descripción en detalle se encuentran en la sección 4.2, fuera de este capítulo por lo fuertemente relacionado que está su funcionamiento con la gestión de información semántica.

- **Planificador semántico**. Este módulo, también llamado en ocasiones *módulo de razonamiento*, es el núcleo de la navegación. Es el que decide qué es lo que se va a hacer para alcanzar un objetivo. Tal vez sea dirigirse a un lugar donde previamente se había percibido un objeto o tal vez sea ir a una estancia concreta donde hay mayor probabilidad de encontrar el objeto deseado, sin descartar otras opciones. No obstante, la "inteligencia"de este comportamiento radica en el empleo que este módulo hace del sistema de inferencia. En cualquier caso, es el módulo que decide lo que va a hacer el robot y a dónde se tiene que dirigir, por lo que se comunica con el resto de módulos del sistema para coordinarlos y gestionarlos.

- **Explorador**. Este componente del sistema se encarga de toda la parte relacionada con el proceso de unir la jerarquía espacial con la jerarquía conceptual, es decir, de filtrar la información sensorial y explorar el mundo para extraer de él la información útil del entorno y usar esa información para lo que sea necesario. Esto en la práctica hace que el explorador se ocupe de tres tareas esenciales. Una es la de ubicar en el mapa del navegador de bajo nivel los objetos que detecta el módulo de percepción. Otra tarea es la de identificar y clasificar el lugar en el que se encuentra en robot, en función de los objetos percibidos. Y la última es la de tomar el control del desplazamiento del robot para facilitar la detección de un objeto determinado.

- **Módulo de percepción**. Todo robot que navegue necesita de la habilidad de percibir el entorno. Sin embargo, en el caso de la navegación semántica, se requiere ir un paso más allá y no limitarse a la detección de obstáculos y espacios libres. Se hace imprescindible que el robot pueda clasificar los lugares y, en este

caso, se le da gran importancia a la clasificación de lugares en función de los objetos que contienen. Por ello se incluye este módulo que en esencia cubre la tarea de agregar todos los sistemas de detección de objetos utilizados en esta tesis.

- **Módulo interactivo**. Este elemento se ocupa de permitir la comunicación entre el usuario y el robot. Es, por lo tanto, una interfaz para el usuario mediante la cual dar órdenes al robot y éste a su vez, pedir instrucciones o información al usuario. En esta tesis se han empleado dos medios diferentes de comunicación. Uno es una interfaz por teclado, recibe órdenes a través de una consola de comandos y muestra los mensajes del sistema por pantalla. El otro consiste en integrar un sistema basado en diálogo y reconocimiento de voz como interfaz con el usuario.

- **Navegador de bajo nivel**. La navegación semántica decide a dónde tiene que ir el robot para cumplir un objetivo. Es en cierta manera añadir una capa de mayor abstracción a la tarea de navegación, sin necesidad de profundizar en el sistema que finalmente lleva al robot a un punto determinado. Por lo tanto, la navegación semántica se puede utilizar sobre un navegador de menor nivel de abstracción como puede ser uno geométrico o topológico. Para ello, en esta tesis se ha diseñado este módulo, que implementa una manera genérica de utilizar inmediatamente la navegación semántica sobre cualquier navegador geométrico o topológico que esté implementado en ROS.

3.2. Planificador

El planificador es la parte de un navegador encargada en decidir la ruta que debe seguir el robot para llegar a su destino. Es el que toma las decisiones de desplazamiento y genera por lo tanto una trayectoria que sigue el robot. En el caso de este planificador que forma parte de un navegador semántico, sus funciones son las mismas. Sin embargo, la trayectoria es en realidad una representación de los destinos semánticos que el robot va encadenando hasta llegar a su objetivo. El destino

semántico es interpretado como una posición que puede ser reconocida en el paradigma de la navegación semántica. En este sistema se reconocen posiciones que contienen objetos específicos, donde se puede realizar una determinada acción o que evocan un sentimiento subjetivo concreto (por ejemplo un lugar tranquilo). Todo eso distingue una posición de otra y se han considerado posiciones semánticas, por lo que pueden ser destinos semánticos. El planificador semántico es el que decide a qué puntos del espacio debe digirse el robot, según el valor semántico que tengan, para cumplir con el deseo del usuario. Por esta responsabilidad de decisión, en esta tesis a menudo se le ha llamado *módulo de razonamiento*. El cumplimiento de sus funciones conlleva que este módulo deba comunicarse con el resto de módulos para coordinarlos, gestionarlos y centralizar en varias ocasiones el flujo de información entre ellos.

Entre otras cosas, este módulo recibe las peticiones del usuario a través del módulo interactivo, consulta los objetos o lugares a los que debe dirigirse mediante su razonador, solicita al explorador que localice e identifique dichos lugares y emite una orden al módulo del navegador de bajo nivel para que llegue hasta ellos.

3.2.1. Componentes del planificador

En este apartado se describen los componentes del planificador, vistos con el diseño de clases de la arquitectura del módulo. Esto facilita la comprensión de las funcionalidades del planificador.

La arquitectura y los elementos que forman el planificador semántico son mostrados mediante el diagrama de clases de la Figura 3.2. Las clases empleadas son las siguientes:

- Planificador semántico (Módulo de razonamiento). Es en realidad el módulo que inica el nodo ROS del planificador y el núcleo de la navegación semántica, por lo que es también el módulo más extenso. Para facilitar su implementación, se han creado el resto de clases que se comentan en este apartado. La función más importante es la de traducir al navegador de bajo nivel una orden de destino dada por el usuario, por lo que debe ser capaz de interpretar la orden semánticamente. El funcionamiento de este método se explica más detalladamente en la sección 3.2.3.

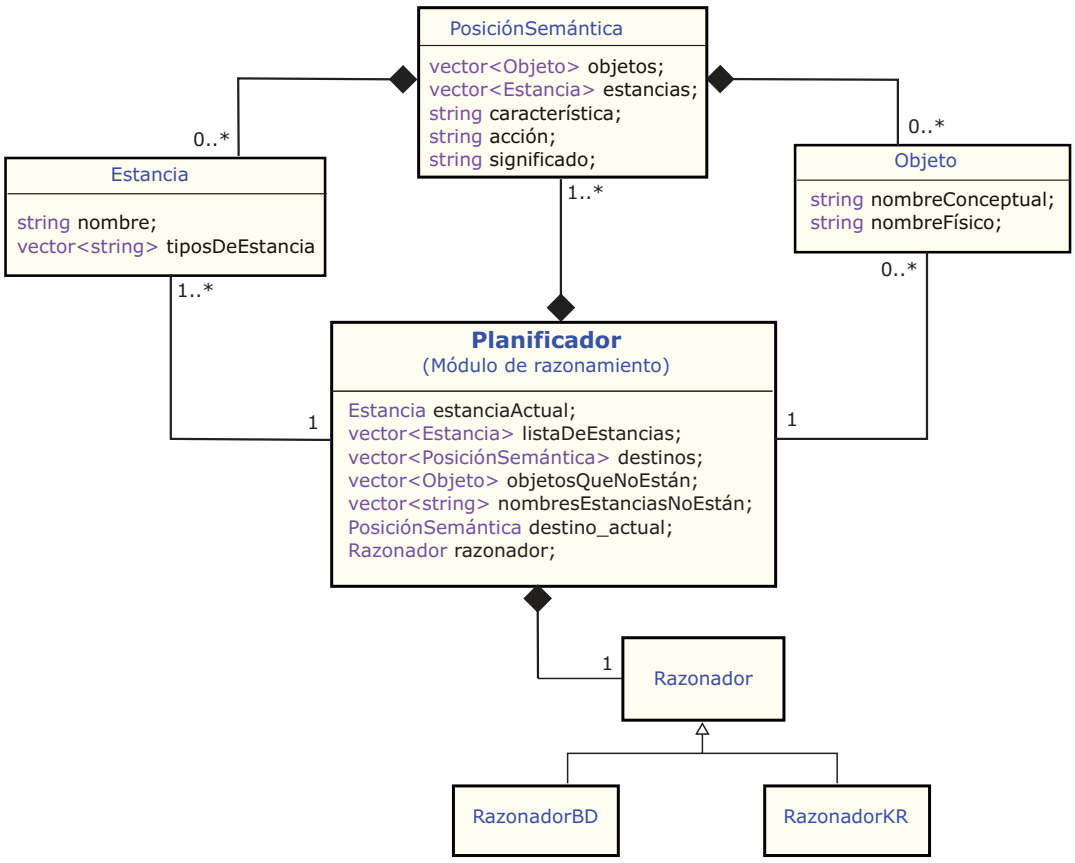


Figura 3.2: Diagrama de clases que forman el planificador semántico.

- Estancia. Esta clase se crea para representar el concepto de una estancia entendida como un lugar delimitado o no por elementos físicos que se diferencia de otro lugar. Si una vivienda tiene 3 dormitorios, 1 salón-comedor, 1 cocina y 2 baños; el sistema tendría 7 estancias. Los atributos de esta clase son el nombre, para identificar la estancia, y un vector de tipos de estancia. Esto es así porque en este sistema se considera que una misma estancia puede estar identificada con muchos tipos de estancia. El ejemplo anterior de salón-comedor es una muestra de ello, donde al mismo lugar físico se le asocian dos tipos de estancia.

- Objeto. Esta es la clase que reprenta los objetos, es decir, las entidades que existen en las estancias. Pueden ser artefactos (hechos o alterados por el hombre

de alguna manera) o no, aunque los ejemplos de objetos contemplados en este sistema de navegación que no son artefactos son muy escasos. Esto es porque esta navegación ha sido probada y desarrollada en entornos domésticos de interiores. Uno de los pocos ejemplos de objetos contemplados que no son artefactos es el agua. En cualquier caso no hay distinción de tipos de objetos en ese sentido. Lo importante de esta observación es que se asume que los objetos del entorno tienen utilidad (por la definición de artefacto) y sirven para algo. Por lo tanto se puede establecer una relación entre utilidad y objeto, y a su vez entre objeto y estancia, con lo que al final se puede relacionar estancia con utilidad. Si el sistema de navegación estuviese pensado para exteriores, sí habría objetos (no artefactos) sin utilidad, como una piedra que no tiene ningún uso ni está en ese lugar por alguna razón. En exteriores, por lo tanto, se habría tenido que realizar una distinción entre objetos y artefactos para establecer las relaciones entre utilidades y lugares.

- PosiciónSemántica. Esta clase se emplea en varios módulos de la arquitectura del sistema. Se ha creado con la intención de facilitar la representación de una posición semántica. El usuario puede solicitar ir a un destino de varias maneras diferentes, y el destino debe ser considerado como una posición objetivo. Las maneras de entender una posición que se han considerado son las que se representan en los atributos de esta clase. Un destino pedido por el usuario puede ser ir a un objeto físico, por lo tanto la posición semántica debe contemplar una lista de objetos presentes en esa posición. Otro destino puede ser ir a una estancia concreta, por lo que la posición semántica debe conocer el tipo de estancia que cataloga el lugar en el que se encuentra. Como se describe antes, un único lugar puede corresponder a varios tipos de estancia, por lo que hay un vector para almacenar todos los tipos a los que corresponde esa posición. Además, los objetos pueden tener alguna característica que cambiaría la posición semántica. Si el usuario quiere encontrar la tostadora de pan sin gluten, dicho objeto tiene que tener la característica de "sin gluten" y es una posición semántica diferente respecto a estar donde hay una tostadora de pan de harina de trigo. Además, la posición también puede estar relacionada con una acción. Si, por ejemplo, la acción es "jugar" quizás sea porque la posición semántica se corresponde con

una sala de juegos de mesa. Por último, otro concepto que se considera en la posición/destino semántico es la capacidad de incluir una valoración subjetiva sobre ese lugar o actividad. En esta tesis se ha llamado *significado*. Como ejemplos de este significado se puede decir que una posición puede tener el *significado* asociado de relajante, divertido, estresante, ruidoso, etc.

- RazonadorBD. El planificador necesita al sistema de inferencias para trabajar con las relaciones conceptuales de la ontología y deducir los destinos que incluye en la planificación. Este sistema de inferencia se compone de un objeto de la subclase Razonador. La primera versión utilizada se ha implementado con un base de datos relacional. Los detalles del sistema de inferencia se comentan en la sección 4.

- Razonador. Es la superclase que forma parte del sistema de inferencia. Contiene todos los métodos que necesita el planificador y obliga a que sus subclases los implementen.

Además de las clases mencionadas que forman el planificador, éste contiene una serie de variables globales que se van a comentar a continuación, puesto que facilitan el entendimiento del proceso de traducción de destino. Estas son:

- **Estancia** estanciaActual. Es la variable donde se almacena la estancia actual donde se encuentra el robot.

- **vector**⟨**PosicionSemantica**⟩ destinos. Este vector contiene la planificación de destinos que va siguiendo el robot hasta que el usuario confirma que se ha cumplido el objetivo. Se va actualizando constantemente y guarda la información tanto de los destinos en los que se ha estado como de los destinos alternativos que deben alcanzarse si falla el destino actual. Por ejemplo, si el usuario quiere ir a un lugar divertido y en la base de conocimiento se encuentra la información que dice que *jugar* y *leer* es divertido, el planificador guarda como destinos: un lugar divertido, jugar y leer. Además, pregunta al razonador por los objetos que sirven para leer y jugar. El razonador podría decir, por ejemplo, que un libro sirve para leer y que la consola de videojuegos sirve para jugar. Entonces el vector de destinos se actualizaría y contendría los destinos: divertido, jugar,

videoconsola, leer y libro. Además, el planificador querría averiguar dónde están los objetos libro y videoconsola. Una respuesta típica podría ser en el salón para ambos, así pues el vector de destinos contendría la información relativa a divertido, jugar, videoconsola, salón (con videoconsola), leer, libro y salón (con libro).

- **vector⟨Objeto⟩** objetosQueNoEstan. En este vector se almacenan los objetos conceptuales que el robot trató de localizar en búsquedas anteriores sin éxito. Se asume que no están en el entorno y se evitan para siguientes búsquedas. Si más tarde el robot explorando detecta un nuevo objeto físico que corresponde con un objeto conceptual que estaba en este vector, es extraído para que pueda ser encontrado en siguientes búsquedas.

- **vector⟨string⟩** nombresEstanciasNoEstan. Esta variable tiene una utilidad análoga a la variable anterior pero con estancias en vez de objetos. Si hay un tipo de estancia que no está en el entorno, ya no se buscará por ella y automáticamente se considerará fallida en siguientes iteraciones.

- **PosicionSemantica** destino_actual. Esta variable contiene el destino actual al que intenta acceder el robot. Se entiende también que puede ser un destino parcial. En el ejemplo anterior cuando se ha comentado el vector de destinos, se ha mencionado el destino absoluto "ir a algún sitio divertido", pero un destino parcial dentro de esta orden sería ir al salón, por ejemplo.

- **RazonadorBD** razonador. Esta variable es el razonador que utiliza el planificador cada vez que necesita realizar una inferencia.

3.2.2. Flujo de topics y servicios del planificador

Debido a que el planificador es el núcleo de la navegación semántica, su comunicación con el resto de módulos es esencial. Requiere un flujo constante de información de distintos módulos. Dicho flujo se representa esquemáticamente en la Figura 3.3. Las flechas bidireccionales rojas corresponden con servicios de ROS mientras que las flechas unidireccionales azules representan un subscriptor y un publicador de ROS en un topic.

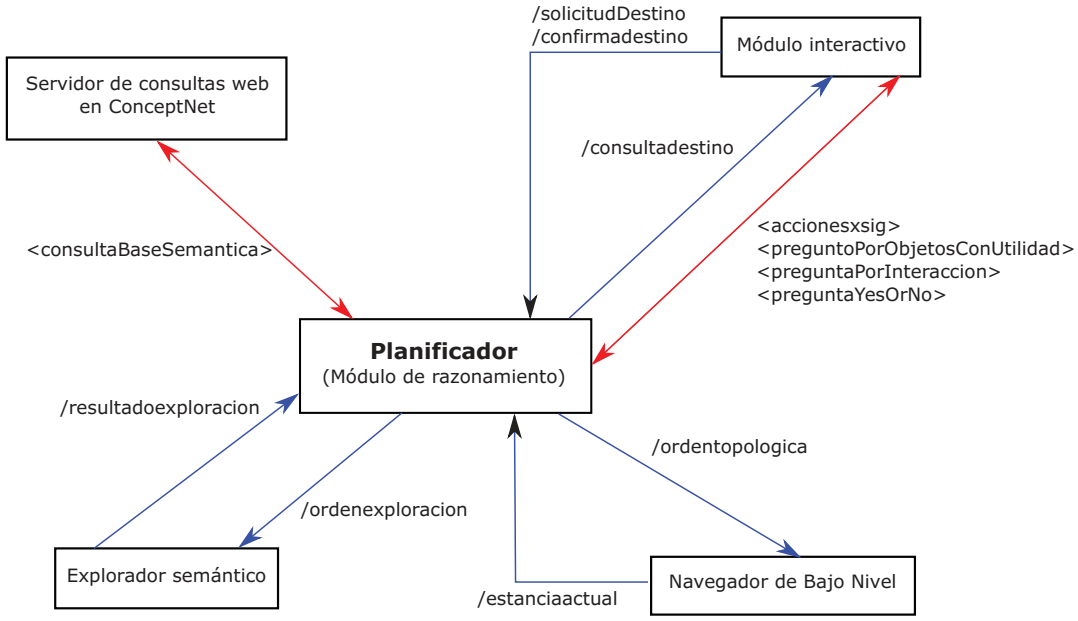


Figura 3.3: Comunicación del Planificador Semántico con el resto de módulos mediante flujo de topics y peticiones de servicios ROS.

Comunicación con el módulo interactivo

El planificador es el módulo que inicia toda esta maquinaria para llevar al robot a un destino, pero el primer punto, que sería recibir la orden, lo hace el módulo interactivo. Por ello, es el primero que debe comunicar con el planificador. Además de recibir la petición de destino por parte del usuario, el módulo interactivo proporciona las funcionalidades necesarias para que el sistema pregunte y se comunique con el usuario cuando sea necesario. Para esta comunicación, el módulo interactivo ofrece los siguientes servicios de ROS:

- $\langle AccionesxSig \rangle$. En este servicio, el cliente (planificador) solicita al módulo interactivo que pregunte al usuario por acciones que tengan un significado concreto. Por ejemplo, para averiguar qué acciones son *divertidas*.

- $\langle preguntoPorObjetosConUtilidad \rangle$. Este servicio se invoca cuando el planificador necesita que el módulo interactivo pregunte al usuario por los objetos

que cumplen una utilidad. Por ejemplo, para saber qué objetos sirven para *descansar*.

- $\langle preguntaPorInteraction \rangle$. En este servicio se atiende a la petición del planificador cuando quiere que el módulo interactivo pregunte al usuario por la interacción que tiene un objeto con otros del entorno. Por ejemplo, si la impresora interacciona con otro objeto. La respuesta podría ser que sí, que la impresora interacciona con el ordenador.

- $\langle preguntaYesOrNo \rangle$. Hay distintas ocasiones, dependiendo del contexto, en el que el planificador necesita una respuesta de SÍ o NO por parte del usuario. Para estas ocasiones se utiliza este servicio.

Además de estos servicios, entre el planificador y el módulo interactivo existe el siguiente flujo de topics:

- /solicitudDestino. Cuando el usuario pide al sistema que vaya a algún lugar, el módulo interactivo recoge esta petición y la publica en este topic para que la reciba el planificador.

- /ConsultaDestino. Cuando el navegador llega a uno de los destinos parciales, el planificador necesita asegurarse de que ese destino es el destino inicial y que por lo tanto el objetivo del usuario está satisfecho. En este topic se publica dicho destino parcial para que lo recoja el módulo interactivo y lance la pregunta al usuario cuando estime oportuno.

- /confirmaDestino. Este topic es publicado por el módulo interactivo y contiene la respuesta del usuario ante la situación del punto anterior.

Comunicación con el explorador semántico

El planificador necesita interactuar con el explorador semántico como se explica con más detalle en la sección 3.3. Los topics mediante los que se establece la comunicación entre módulos son:

- /ordenexploración. Este topic es publicado por el planificador y lo recibe el explorador. Incluye un código de orden que interpreta el explorador y de este modo sabe lo que tiene que hacer.

- /resultadoexploración. Este topic es publicado por el explorador y lo recibe el planificador. Mediante esta vía, el explorador comunica el resultado de la orden que había recibido.

Comunicación con el navegador de bajo nivel

El módulo navegador de bajo nivel, que se explica con detalle en la sección 3.6, recibe las órdenes del planificador semántico. Los topics involucrados en esta comunicación son:

- /ordentopologica. En este topic se incluye también cualquier destino geométrico. Es donde el planificador publica el punto o nodo donde el navegador de bajo nivel debe llevar al robot.

- /estanciaActual. En este topic el navegador de bajo nivel mantiene actualizado al planificador de lo que sucede en su ámbito. Aquí publica la estancia actual en la que se encuentra y con esa información el planificador también conoce el resultado de una orden previa.

Comunicación con el nodo de consulta en web semántica

En el sistema se ha incluido un nodo que realiza ciertas consultas en una web semántica. El objetivo de esto es trascender el conocimiento local que tiene el robot sobre el entorno dentro de lo posible, de este modo se vuelve más autónomo. La manera en la que se aplica está relacionada con el proceso de adquisición de nuevo conocimiento. Cuando el planificador necesita obtener nueva información para que el razonador tenga más capacidad de inferencia, pero es información que no puede aportar el sistema sensorial, sólo tiene dos opciones. La primera que se implementó es la opción de preguntar al usuario, como se ha visto en esta sección. La segunda es probar en primer lugar si la información aparece en una web semántica que ofrece

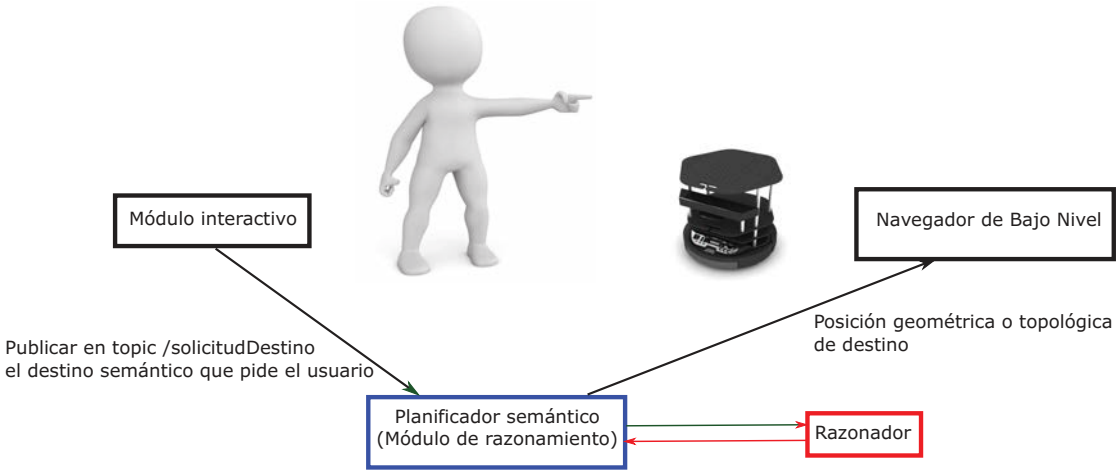


Figura 3.4: El planificador tiene como principal función traducir el destino semántico en un destino entendible por el navegador de bajo nivel.

servicios de consulta de información para sistemas de inteligencia artificial y robots. Si la información no se consigue en la consulta web, entonces sí se pregunta al usuario.

El nodo implementado ofrece un servicio ROS mientras que por otro lado accede a un servicio web. El servicio ROS, que es el medio de comunicación con el planificador semántico y, por tanto, con el sistema de navegación es $\langle consultaBaseSemantica \rangle$. Este módulo se explica con más detalle dentro del capítulo 5.

3.2.3. Cálculo recursivo del destino.

El planificador es el encargado de traducir el destino semántico que pide el usuario en un destino entendible por el navegador de bajo nivel. La figura 3.4 ilustra este proceso que se describe con más detalle en la Figura 3.5.

Inicialmente, se parte de una petición del usuario para alcanzar un objetivo. Esa petición es capturada por el módulo interactivo, el cual recoge el destino semántico que ha solicitado el usuario. Este destino puede ser ir a un sitio tranquilo (búsqueda por significado *tranquilo*), ir a ver la televisión (búsqueda por objeto *televisión*), ir a la cocina (búsqueda por estancia *cocina*), ir a beber algo caliente (búsqueda por acción *beber* con característica *caliente*), ir a descansar (búsqueda por acción *descansar*), etc.

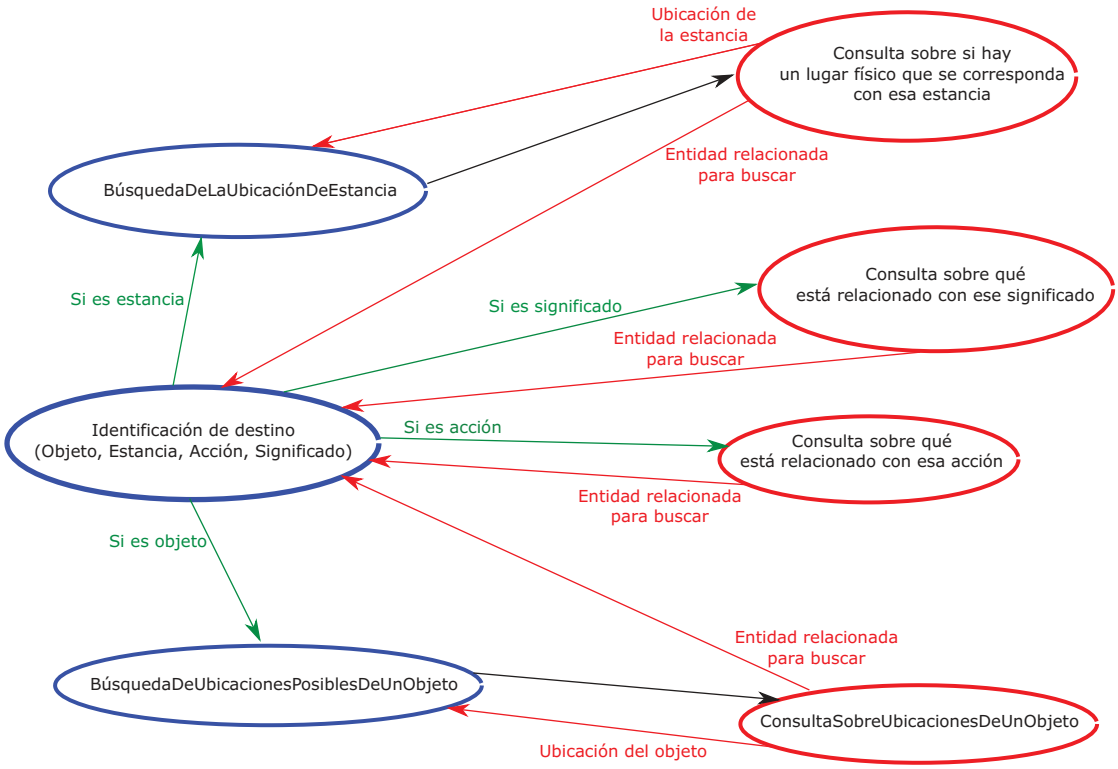


Figura 3.5: Esquema del funcionamiento del proceso de traducción de destino semántico que se lleva a cabo en el planificador. Las elipses azules representan funciones del propio planificador, mientras que las elipses rojas representan funciones que se ejecutan dentro del razonador.

La petición de destino es recogida por el planificador semántico, el cual identifica el tipo de destino e inicia una secuencia de consultas al razonador hasta que obtiene un lugar u objeto concretos a los que dirigirse que se correspondan con una posición entendible por el navegador de bajo nivel (una coordenada o un nodo). Esta secuencia puede ser un proceso recursivo como se aprecia si se observa detenidamente la Figura 3.5. El método que recibe en primera instancia el destino semántico es el de *Identificación de destino*. Aquí se comprueba el tipo de destino que se ha solicitado, pues requieren tratamientos diferentes.

- Pregunta por objeto. Si el destino es un objeto, se llama a un método que trata de averiguar la ubicación posible de ese objeto. Este proceso es resuelto mediante

una consulta al Razonador. Hay dos tipos de respuesta posibles, dependiendo de la información de la que se dispusiera. Si, por ejemplo, el objeto es el ordenador y el sistema había percibido un ordenador en el entorno, la respuesta de la consulta es el ordenador percibido, con su ubicación. Pero también puede ser que no se haya percibido ningún ordenador y el Razonador tenga que ofrecer destinos parciales. Entonces la respuesta podría ser, para el caso del ordenador: *despacho*, *impresora*. Es decir, responde con las entidades que están relacionadas con el ordenador y cuya búsqueda parcial puede acercar al robot al ordenador. Como se había adelantado, esta búsqueda del destino es recursiva, puesto que el sistema trata de resolver los destinos parciales (en este caso buscaría de nuevo pero esta vez por estancia -despacho- y por objeto -impresora-), llamando a la misma función de *Identificación de destino* pero con los destinos parciales que están asociados al destino principal.

- Pregunta por estancia. Si el destino es una estancia, el método llamado es el encargado de obtener las ubicaciones posibles de dicha estancia. De forma análoga al caso anterior, el método utiliza al Razonador para obtener las posibles localizaciones de un tipo de estancia. Por ejemplo, tal vez la cocina sea la estancia-2. También puede darse el caso de que el Razonador no encuentre una relación directa entre el tipo de estancia y una estancia fisica del entorno. En ese caso la respuesta será otra entidad que pueda estar relacionada con el tipo de estancia solicitado.

- Pregunta por acción. Cuando el destino es una acción, por ejemplo *descansar*, el planificador llama a un método que invoca al Razonador para intentar encontrar una entidad relacionada con esa acción. A diferencia de los casos anteriores, en esta ocasión no hay más alternativas puesto que las acciones no se relacionan directamente con puntos en un mapa de bajo nivel, sino con objetos del entorno.

- Pregunta por significado. El caso en el que el destino es un significado lleva a un comportamiento similar al caso de la pregunta por acción. El planificador llama al método correspondiente que utiliza al Razonador para obtener destinos parciales que estén relacionados con dicho significado. Pero no habrá nunca una

respuesta que sea un lugar del mapa de bajo nivel porque los significados se asocian con acciones.

El proceso de cálculo de destino termina cuando se han obtenido los destinos parciales o destino absoluto y una vez finalizado, se envía esa posición al navegador de bajo nivel. Si el usuario indica que ese no es el destino que él estaba buscando o hay algún fallo en la localización del destino actual, el sistema repite el proceso de cálculo de destino eliminando los lugares descartados por una u otra razón. Y continuará con este comportamiento hasta que el usuario indique que el destino es el correcto o haya explotado todas las opciones, en cuyo caso indica que la búsqueda ha fracasado y solicita al usuario más información o que desista de llegar a ese objetivo.

De este modo queda descrito esquemáticamente el funcionamiento del planificador y de la búsqueda recursiva del destino. En el capítulo 4 se explica con más profundidad el mecanismo de inferencia y de obtención de destinos.

3.3. Explorador

El módulo explorador se encarga de procesar la información sensorial que se recibe del agregador de percepciones. Extrae la información semántica que aportan los objetos percibidos. Esta información del exterior se utiliza para dos objetivos concretos:

- **Ubicar los objetos en el mapa del navegador de bajo nivel**. El sistema necesita situar en el mapa del navegador de bajo nivel los objetos percibidos del mundo real. Cuando el navegador semántico está construido sobre un navegador geométrico, el explorador toma la coordenada relativa del objeto, la transforma en una coordenada absoluta del mapa y almacena tanto la posición absoluta del objeto como la posición del robot desde donde el objeto fue percibido. Si el navegador semántico está construido sobre un navegador topológico, el explorador identifica el nodo en el que se encuentra y asocia el objeto a dicho nodo.

- **Clasificar la estancia en la que se encuentra el robot**. Cuando el robot detecta un objeto que identifica un tipo de estancia, el explorador semántico se encarga de que se asocie ese tipo de estancia con el lugar en el que se encuentra. La ubicación del robot en el momento en el que el objeto diferenciador

es detectado, se asocia con la posición de la estancia. Esto se hace del mismo modo para un navegador geométrico, guardando la posición actual, como para un navegador topológico, guardando el nodo actual. Entonces, el robot se encuentra en la estancia 1 cuando detecta la presencia un objeto que significa que el tipo de estancia de ese lugar es A, y el robot se encontraba en la posición B, el explorador guarda que la estancia 1 es de tipo A y además, la posición de la estancia es B.

Además, el explorador también tiene la responsabilidad de gestionar el resto del sistema para localizar un destino desconocido. Por ejemplo, para localizar un objeto determinado cuando el navegador semántico ya ha llevado al robot al lugar con más probabilidad de encontrar dicho objeto.

3.3.1. Estructura y diseño del explorador

El explorador semántico se comunica con el resto del sistema de navegación de la manera que se puede ver en la Figura 3.6. Recibe la información de los siguientes topics:

- **<perceptionData>**. Este topic es emitido por el agregador de percepciones, en forma de un único tópic donde se vuelca la información de todos los objetos percibidos, independientemente de la tecnología empleada para realizar esa detección. La recepción de este topic conlleva la ejecución de la función más destacada del explorador: PerceptionAction, cuyo funcionamiento es explicado en la sección 3.3.3.

- **<odom>**. Este topic es el que indica la posición actual del robot según el navegador geométrico. Para navegadores topológicos el topic será diferente. La recepción de este topic llama a una función que hace que el explorador actualice una variable interna donde se guarda la posición actual del robot.

- **<ordenExploración>**. Este topic es la vía en la que el explorador recibe órdenes de parte del planificador. Estas órdenes cambian el estado del explorador, tal como se ve en la Figura 3.7.

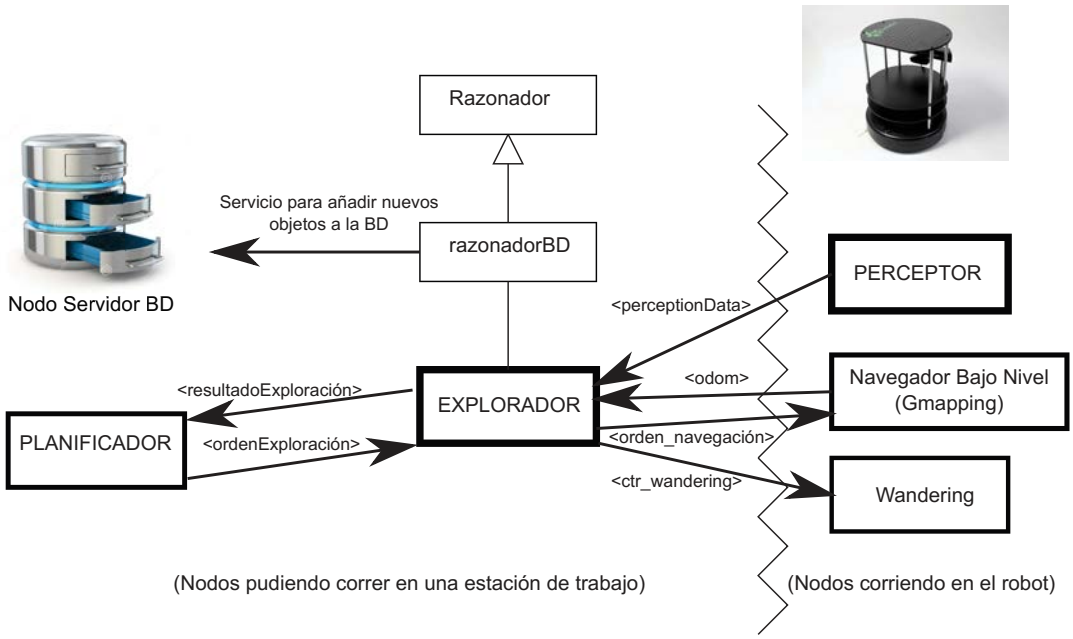


Figura 3.6: El módulo explorador y su flujo de información en topics con el resto de módulos.

Además, el explorador semántico emite la siguiente información en forma de topics:

- **<orden_navegación>**. El explorador semántico emite órdenes al navegador de bajo nivel a través de este topic. La situación que lleva a la emisión de una orden por parte del explorador es que esté en un estado de búsqueda de objeto o de destino y haya recibido por parte del agregador de percepciones la información de que el objeto deseado ha entrado en el campo visual del robot. La orden que emite el explorador en este caso es de que el robot se aproxime al objeto detectado, iniciando una navegación geométrica relativa.

- **<ctr_wandering>**. Debido a que el explorador será el módulo llamado cuando se quiera encontrar un objeto desconocido, se requiere que este módulo tenga control sobre la función *wandering* que está en el robot. Esta función no ha sido desarrollada en esta tesis, se obtuvo de un repositorio gratuito y sirve para hacer que el robot deambule sin rumbo fijo de forma aleatoria. Hay dos situaciones que producen una activación de esta función de deambular. Una de

ellas es cuando el robot recibió la orden de buscar un objeto del cual no se conoce que exista una instancia en el mundo real, pero el módulo razonador sí conoce cuál es la estancia más idónea para realizar la búsqueda. En este caso, el navegador de bajo nivel lleva al robot a esa estancia y entonces el control pasa al módulo explorador, el cual activa la función *wandering* y se detendrá cuando encuentre el objeto o se agote un tiempo prefijado. La otra situación es cuando o bien partimos de la situación anterior pero el objeto no fue detectado en el tiempo estipulado o bien desde el primer momento el planificador no supo a qué estancia llevar el robot para iniciar la búsqueda. En este caso el explorador activa la función *wandering* con esperanza de que recorra cualquier estancia del entorno y para ello cuenta con un tiempo predefinido mayor.

- **<resultado_exploración>**. El explorador comunica el resultado de la exploración a través de este topic, en el cual publica un mensaje de confirmación tanto si encontró lo que se le había pedido como si saltaron los timeout sin éxito en la búsqueda. En cualquier caso, el resto del sistema sabe que la recepción de este topic implica que la tarea de exploración ha finalizado.

Es importante observar que el explorador tiene un objeto de la clase *RazonadorBD*, la cual hereda de la clase *Razonador*. Esto es porque un explorador semántico requiere de la capacidad de inferencia de un razonador en tareas como la clasificación de estancias. Además, el explorador tiene que tener la capacidad de añadir información a la base de datos de objetos físicos, pues una de sus tareas principales es la detección y posicionamiento de objetos detectados. Por lo tanto, los métodos del objeto RazonadorBD que el explorador utiliza son:

- **anadirObjetoFisico(nombreObjeto, nombreEstancia, posicionObjetoEnMapa, posicionRobot, nodoActual)**
  Este método es el correspondiente a añadir un nuevo objeto físico detectado por el robot. Donde los argumenos son, respectivamente, el nombre del objeto conceptual identificado, el nombre de la estancia actual, la posición del objeto en el mapa, la posición actual del robot en el mapa y el identificador numérico del nodo asociado a la posición actual del robot.

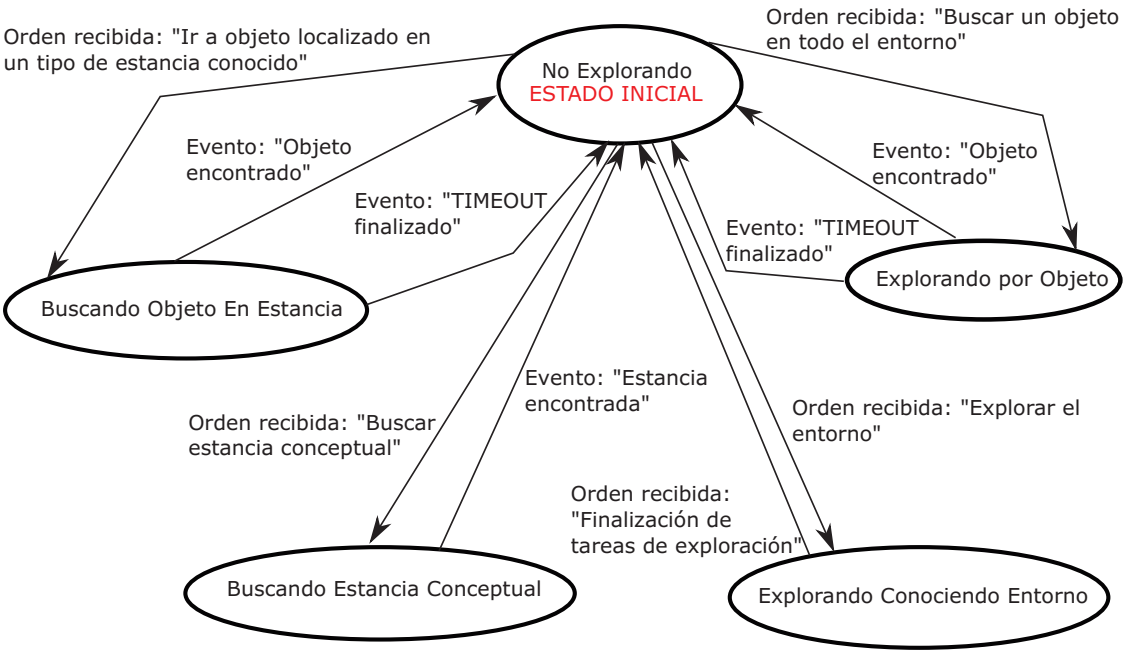


Figura 3.7: Diagrama de estados del módulo explorador.

- **identificarEstanciaFisica(nombreObjetos,estanciaActual)**
  Este método es el llamado para clasificar la estancia actual en función de los objetos que se perciben. Los argumentos son la lista de objetos percibidos y la estancia actual donde se encuentra el robot.

3.3.2. Estados del explorador

La relativa complejidad del explorador requiere que su funcionamiento sea gestionado por una serie de estados que se muestran en la Figura 3.7. Inicialmente el estado inicial es el estado NO EXPLORANDO. En función de las órdenes recibidas por el planificador, el explorador puede pasar a varios estados:

Buscando objetos en estancia. A este estado se llega mediante una orden recibida desde el planificador cuando éste quiere que el sistema encuentre un objeto cuya localización se conoce o se prevé. De este estado se vuelve al estado inicial tanto si se encontró el objeto como si se agotó el tiempo establecido.

Explorando por objeto. El sistema llega a este estado cuando no se ha encontrado un objeto en la estancia en la que se suponía que iba a estar. Entonces se amplia la búsqueda a todo el entorno. Otra manera de llegar a este estado es cuando inicialmente no se conoce el mejor lugar para empezar una búsqueda por un objeto y directamente se decide buscarlo por todo el entorno. Al igual que en el caso anterior, el éxito buscando el objeto o la finalización del tiempo establecido llevan al estado inicial.

Buscando estancia conceptual Este estado se activa cuando el sistema requiere una búsqueda de un tipo de estancia conceptual concreta. El robot explorará por todo el entorno hasta que encuentre algún objeto que identifique la estancia actual como la estancia deseada o se agote el tiempo establecido. En ambas situaciones ser vuelve al estado inicial.

Explorando conociendo el entorno Cuando el sistema requiere que se elabore un mapa, el explorador continua funcionando recibiendo la información de los objetos que percibe. Si está en este estado se limita a guardar dichos objetos y clasificar la estancia si procede. Se sale de este estado cuando una orden del planificador devuelve al explorador al estado inicial.

3.3.3. Acciones del explorador

En la implementación del explorador se ha tenido en cuenta las acciones que debía realizar en función de los estímulos o topics recibidos. Lo más reseñable es la acción que realizan los subscriptores de los topics de recepción de órdenes por parte del planificador y de recepción de objeto detectado.

- Acción por recepción de orden. Esta función actualiza el estado del explorador en función de la orden recibida del módulo de razonamiento (planificador) publicada en el topic <ordenExploración>.

- Acción por recepción de topic de percepción (objeto detectado). El explorador está subscrito al topic donde se publica la información de percepción, por lo tanto cada vez que el robot percibe un objeto, la función subscriptora del explorador es llamada. Lo primero que realiza es la comprobación de que si el objeto que se

está percibiendo es un objeto conocido (visto previamente) o si por el contrario es la primera vez que el robot percibe dicho objeto. Si es un objeto nuevo, será almacenado en la base de datos con su posición relativa al mapa geométrico si la navegación de bajo nivel es geométrica o al nodo correspondiente si la navegación es topológica. Para discernir entre si el objeto es nuevo o ya había sido percibido, lo que se hace es cotejar la posición de los objetos conocidos del tipo de objeto conceptual percibido con la posición del objeto que se está percibiendo en ese momento. Si coinciden se entiende que el objeto ya había sido registrado anteriormente. Por lo tanto, el explorador obtiene la posición del objeto actual en el mapa antes que nada. En una navegación topológica, se limitaría a obtener el nodo actual, en la navegación geométrica hay que realizar una transformación de coordenadas. Esto es porque se necesita la información de la posición absoluta (relativa al mapa) pero en este punto lo que se tiene es la posición absoluta del robot y la posición relativa del objeto (es decir, la posición del objeto en el sistema de referencia del robot, no en el sistema de referencia del mapa). El ángulo de rotación necesario para la transformación de coordenadas se obtiene con el algoritmo 1. Y con este ángulo, se obtiene la matriz de rotación de 3.1 y a las coordenadas rotadas se le suma la traslación del robot en el mapa como se muestra en la ecuación 3.2 y con esto se obtienen las coordenadas absolutas del objeto percibido relativas al mapa. Después de esto, se llama a la función del RazonadorBD que se encarga de añadir un objeto físico a la base de datos. El proceso de diferenciación entre un objeto ya percibido de uno nuevo se ajusta mediante el parámetro MARGEN_SIMILITUD_COORDENADA que indica cuál es la distancia entre un objeto que se está percibiendo respecto a un objeto ya percibido que hace que se les considere diferentes. En caso de considerarse diferente, el nuevo objeto es añadido y su posición es vinculada al mapa.

$$(x_{rotada}, y_{rotada}) = (x_{robot}, y_{robot}) \begin{pmatrix} \cos(angle) & \sin(angle) \\ -\sin(angle) & \cos(angle) \end{pmatrix} \tag{3.1}$$

$$(x_{mapa}, y_{mapa}) = (x_{rotada} + x_{robot}, y_{rotada} + y_{robot}) \tag{3.2}$$

Algorithm 1 Algoritmo de obtención de ángulo de rotación

Require: Posición del robot absoluta *position* (relativa a mapa).

1: $signo = 1$
2: **if** $position.orientation.z < 1$ **then**
3: $\quad signo = -1$
4: **end if**
5: $angle = signo * 2 * \arccos(position.orientation.w)$
6: **return** *angle*

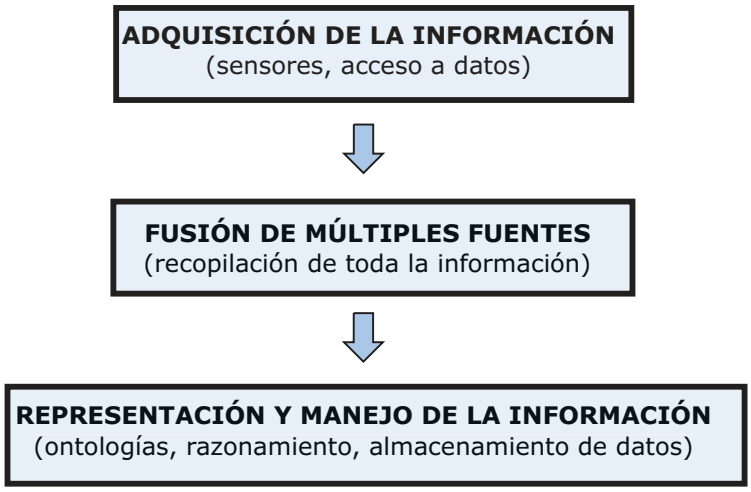


Figura 3.8: Información semántica

3.4. Módulo sensorial

El sistema de navegación semántico requiere de un módulo sensorial que sea capaz de adquirir la información del entorno. Para potenciar la capacidad sensorial, se ha tratado de fusionar la información de distintos subsistemas de percepción. Esto incluye diferente software sobre el mismo dispositivo como (una cámara empleada por distintos clasificadores de objetos) como incluir sensores de otros tipos (como micrófonos). Estos datos luego son presentados a los módulos que lo necesiten para que gestionen la información con ontologías, razonen o simplemente almacenen los datos (Figura 3.8).

El sistema desarrollado se puede calificar como un sistema multisensorial que es aprovechado por la navegación a todos los niveles, los cuales se comunican entre sí

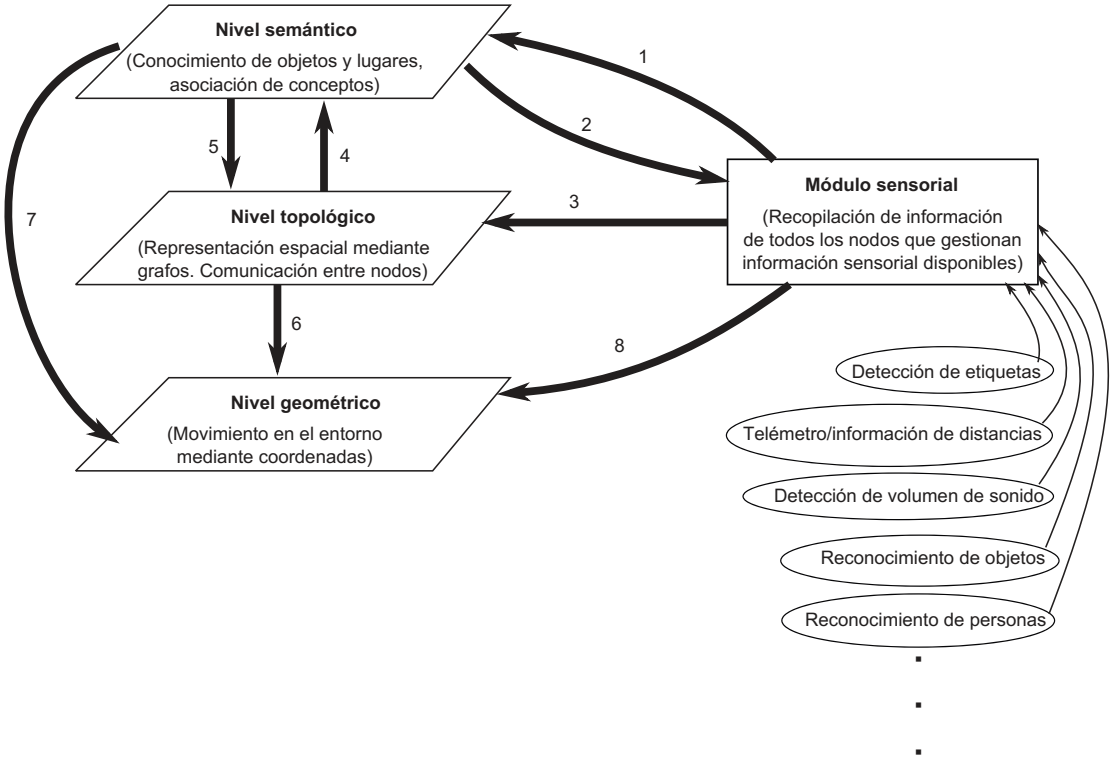


Figura 3.9: Flujo de información entre las diferentes capas del navegador y el módulo sensorial.

para aumentar la eficiencia. El flujo de esta comunicación está descrito en la Figura 3.9. En esta figura las flechas que simbolizan el flujo de información están numeradas, a continuación se procede a comentar cada una de ellas:

1. El nivel semántico de la navegación puede mejorar la propia percepción de los objetos, por ello se permite un flujo de información entre la capa semántica y el módulo sensorial. La razón de que sea posible una mejora viene de que una de las funciones del módulo sensorial es reconocer objetos en el entorno. Este proceso se realiza mediante varios sistemas, que serán expuestos a lo largo de esta tesis, y uno de ellos es un sistema dedicado al reconocimiento de objetos por clasificación de contornos. Este sistema funciona mediante extracción de características visuales mediante una cámara y el posterior aprendizaje de dichas características mediante una máquina de soporte vectorial. La eficiencia y

porcentaje de éxito de este sistema son comentados dentro de la sección 3.4.2, pero una característica que ayuda a reconocer el objeto es precisamente conocer el lugar (a nivel semántico) en el que se encuentre el robot. Esto es así porque distintos objetos pueden tener contornos similares como un televisor y un microondas. Sin embargo, conocer que la ubicación actual es, por ejemplo, una cocina, haría que aumentara la posibilidad de que dicho objeto es un microondas, al contrario de que si la ubicación actual es un salón.

2. Este flujo de información desde el módulo sensorial hasta el nivel semántico de navegación ha sido justificado en distintas ocasiones anteriormente. La navegación semántica necesita la información de los objetos que el robot percibe a su alrededor, para poder identificar el lugar en el que se encuentra con una ubicación conceptual.

3. A su vez, el nivel topológico también necesita la información de lo que se percibe mediante el módulo sensorial. En la navegación a nivel topológico, el robot obtiene una trayectoria que indica los nodos por los que tiene que pasar hasta llegar a su destino. La manera que tiene un navegador topológico de guiarse se basa en percibir eventos, que consisten en detectar los puntos de referencia que al robot le indican el camino para llegar de un nodo a otro. Estos puntos de referencia están limitados por el nivel sensorial y en la mayoría de casos es suficiente con detectar una puerta. Por eso muchos navegadores topológicos pueden funcionar con un láser y odometría. Sin embargo, en este sistema que alcanza un nivel semántico de navegación, se propone un nivel topológico que aprovecha la mayor implicación de las habilidades perceptivas del robot y los puntos de referencia que se emplean en el nivel topológico contemplan grandes objetos del entorno que se pueden considerar fijos.

4. El nivel topológico y semántico funcionan apoyándose el uno en el otro. Esta dirección de flujo de información sirve para que el nivel semántico pueda asociar un concepto a un lugar físico del entorno, y para ello necesita recibir del nivel topológico la información del nodo actual en el que se encuentra el robot.

5. De forma similar a la anterior, la estrecha relación entre el nivel topológico y semántico demanda que exista un flujo de información para que el nivel topológico reciba el nodo objetivo al que dirigirse.

6. La relación entre el nivel geométrico y topológico es necesaria para cubrir casos que requieran un desplazamiento más concreto del robot en el entorno que no esté asociado a un nodo.

7. La relación directa entre el nivel geométrico y semántico surge de nuevo de una situación en la que se llega a un nodo topológico cercano al destino, pero no es exactamente el destino. El sistema responde en este caso explorando hasta encontrar el objeto deseado. El movimiento en esta exploración es controlado por un nivel geométrico.

8. El nivel geométrico también necesita información sensorial del entorno, en su caso de telemetría.

El nivel topológico, geométrico y la navegación, junto con el módulo sensorial cuya función es percibir la información del entorno, experimentan una especie de sinergia en la que cada uno de estos elementos aumenta su eficiencia por la implicación de los otros.

3.4.1. Fusión de sistemas de percepción. Interfaz perceptiva.

A lo largo de todo el desarrollo de esta tesis, uno de los puntos del sistema a mejorar siempre ha sido la detección de objetos. La capacidad sensorial del robot es crítica en la navegación semántica, el éxito depende en gran parte de la capacidad del robot para detectar y clasificar los objetos que percibe. Es por ello que durante esta tesis se abrieron diversas líneas simultáneas de investigación con el objetivo de añadir técnicas de detección de objetos al robot. Algunas de esas líneas de investigación se llevaron en forma de dirección de trabajos finales de máster y luego se trabajó en su integración con el navegador semántico. Por todo esto, se consideró indispensable que el módulo de percepción fuese constituido por un agregador de percepciones (también llamado ocasionalmente interfaz de percepción) de modo que se recopile toda la información relativa a los objetos detectados que aporta cada subsistema de

percepción y se unifique en un único topic donde en última instancia se publican todos los objetos detectados independientemente de la tecnología empleada para detectarse.

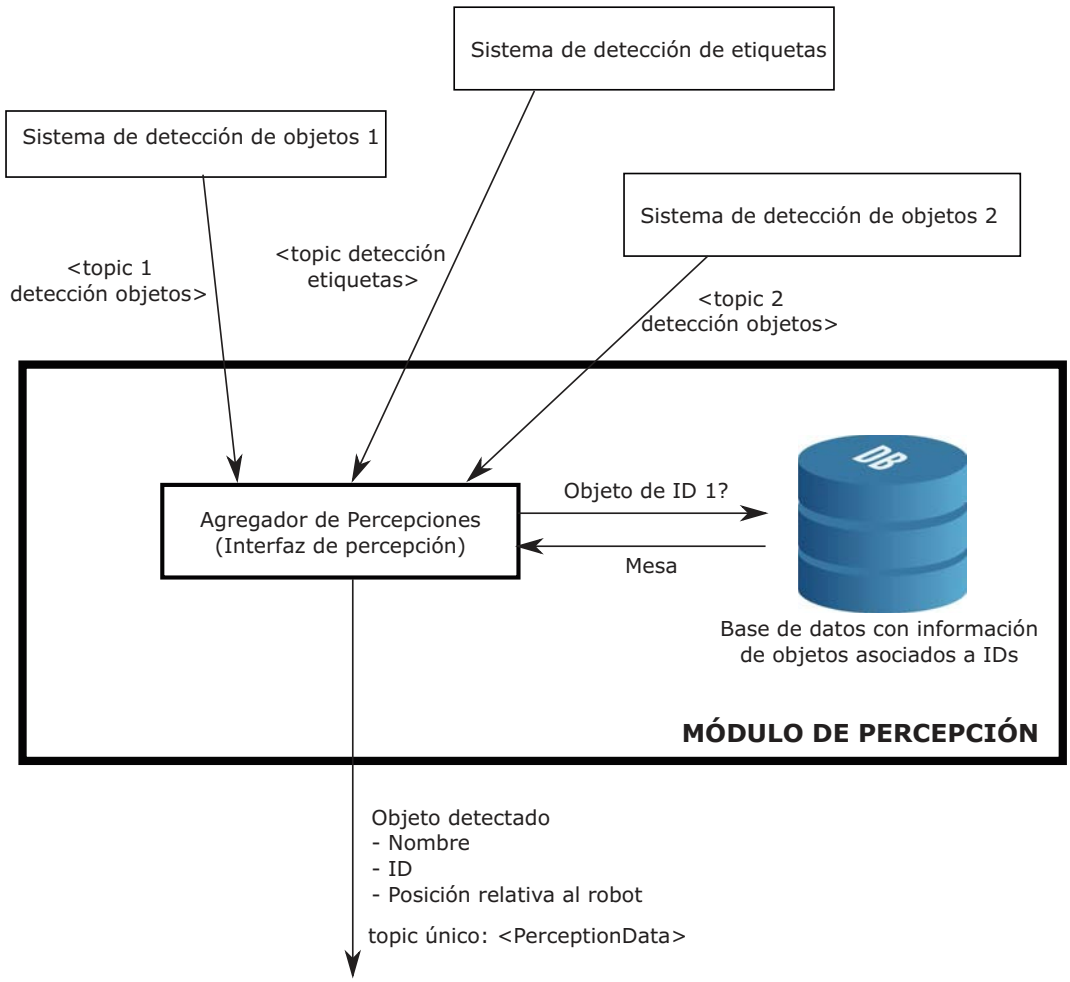


Figura 3.10: Módulo de percepción: Agregador de percepciones.

La Figura 3.10 muestra la estructura del módulo de percepción. El núcleo de dicho módulo es el agregador de percepciones, el cual está subscrito a todos los topics donde publican cada uno de los subsistemas de detección de objetos que se utilizan. El agregador también necesita de una base de datos para obtener la información del objeto detectado cuando la detección de objetos se está realizando mediante etiquetas

o marcas. La información sensorial puede extraerse de tres modos diferentes, según la naturaleza del subsistema de detección:

- **Clasificador visual de objetos.** Son los subsistemas de detección más numerosos en esta tesis, habiéndose empleado hasta tres de ellos. Normalmente constan de un clasificador que es previamente entrenado para reconocer una serie de objetos percibidos con una cámara o kinect. Pueden basarse en detección de contornos y formas, que permite una detección genérica de varios objetos del mismo tipo, o en una detección de objetos concretos mediante descriptores que buscan aquellos píxeles en la imagen que resultan diferenciadores. Son los más complejos y susceptibles a fallos, además de presentar otros problemas como un coste computacional bastante alto. Sin embargo son los únicos que realmente detectan y clasifican un objeto del entorno.

- **Clasificador de objetos etiquetados.** Debido a la dificultad de desarrollar y emplear los clasificadores anteriores, una de las primeras alternativas que se emplearon fue la de sustituir una detección visual de objetos por un reconocedor de etiquetas. Esto requiere añadir una etiqueta al objeto, lo que va en contra de la filosofía de tener que modificar de ninguna manera un entorno humano para que un robot pueda realizar sus tareas en él. Sin embargo permitió avanzar rápidamente en otros temas pues soluciona de una manera simple el problema de la visión artificial y la detección de objetos. Estos sistemas requieren que el agregador de percepciones se comunique con una base de datos donde está la información de los objetos a los que se corresponden los IDs de las etiquetas detectadas. De este modo, si el sistema detecta la etiqueta 1, y la etiqueta 1 corresponde a un ordenador, el sistema entiende que está detectando un ordenador. El sistema de detección de etiquetas utilizado en esta tesis esta basado en ARToolkit [1] y se emplea el paquete de código abierto de ar_pose para ROS y etiquetas cuadradas con configuraciones diferentes.

- **Detección de etiquetas.** Por sí mismo, el sistema de detección de etiquetas puede ser aprovechado para reconocer lugares por el navegador de bajo nivel. En concreto, el navegador topológico utilizado requiere de la habilidad de detectar

[1] http://wiki.ros.org/artoolkit

marcas en el entorno. Se ha empleado el mismo sistema basado en ARToolkit, pero es necesario identificar por separado las dos funcionalidades. Mientras que en el punto anterior se trata de un detector de objetos etiquetados, en este punto la etiqueta no se corresponde con ningún objeto y es simplemente una marca empleada como punto de referencia por el navegador topológico. Para diferenciar con el caso anterior, cuando la etiqueta no está asociada con ningún objeto en la base de datos de consulta no aparece ningún objeto referenciado y de este modo el objeto es justamente la etiqueta con su ID.

Por lo tanto, como puede verse en la Figura 3.10, el módulo de percepción ofrece una única salida, publicando en un topic único al que se subscriben los módulos del sistema que requieren de la información de los objetos percibidos. Y esta información consta de:

- Nombre del objeto. Se refiere al nombre conceptual, a lo que es ese objeto.
- ID del objeto. Esto se emplea cuando se ha reconocido el objeto como un objeto concreto, único en el entorno y por lo tanto identificado con un ID.
- Posición relativa respecto al robot. Es necesario saber dónde está el objeto desde el sistema de referencia del robot para ubicarlo en el mapa.

3.4.2. Tecnologías de detección de objetos

En este apartado se comentan brevemente los distintos sistemas de detección de objetos empleados. Dos de ellos son el fruto de la dirección de sendas tesis de máster y sus resultados fueron publicados en [6] y [55]. El tercer sistema empleado está basado en RoboEarth y se integró en el sistema de navegación, aunque era un algoritmo muy pesado además de orientado a estar embarcado en una cámara estática y en la práctica no resultó útil en el navegador semántico.

Detección con RoboEarth

La primera idea para dotar a la plataforma robótica empleada en esta tesis de la habilidad de detectar objetos, fue incorporar un sistema robusto y abierto de detección

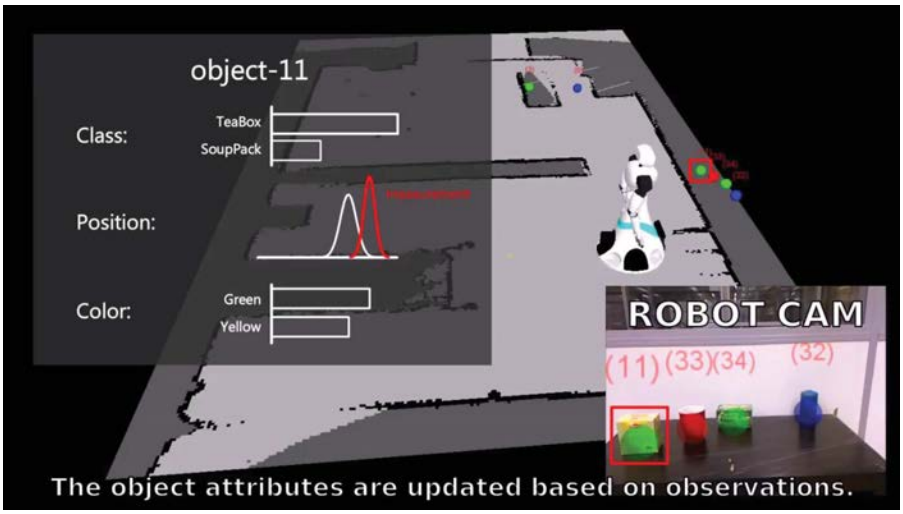


Figura 3.11: Captura de vídeo de ejemplo de detección de objetos con RoboEarth.

de objetos ya desarrollado. Con este objetivo en mente se trató de integrar RoboEarth en el sistema. Dicho proyecto tiene como objetivo probar que un repositorio con una gran cantidad de información accesible por internet aumenta considerablemente los procesos de aprendizaje y de adaptación para realizar tareas complejas. Además, sostienen que dicho repositorio también implica aumentar la capacidad de llevar a cabo de forma autónoma tareas para las cuales no existía una planificación explícita. RoboEarth [2] apuesta por crear un Internet para robots, donde los robots compartan conocimiento entre ellos sobre objetos, entornos y acciones.

El tratar de integrar el proyecto de RoboEarth sirvió como inspiración para intentar aprovechar de un modo similar la información útil para robots en internet (véase la sección 5.4). Y además, se encontró lo que inicialmente se deseaba puesto que parte ese proyecto se centra en sistemas de detección de objetos, con muestras de funcionamiento como se ve en la captura de pantalla de la Figura 3.11 y con amplia información sobre su instalación [3] y ejecución [4].

No obstante, los resultados no fueron lo esperado. El sistema no era muy fiable si se embarcaba en un robot en movimiento y además requería equipos demasiado potentes. En la práctica no se podía acoplar en el turtlebot empleado en esta tesis.

[2] wiki.ros.org/roboearth
[3] http://wiki.ros.org/roboearth_stack
[4] http://wiki.ros.org/roboearth_stack/Tutorials/Usingrecordedmodelsforobjectdetection

Estas pruebas de integración están recogidas en el capítulo de experimentos. Esto llevó a considerar alternativas a la detección de objetos y se llegó a la conclusión de que incluir un sistema sencillo de detección de etiquetas podría ser muy útil, tanto para simular la detección de objetos como para mejorar los futuros sistemas de detección que se consideraron implementar en el futuro inmediato. Los cuales efectivamente se implementaron como se comenta a continuación.

Detección de objetos por análisis de contorno y descriptores

El desarrollo de este método fue realizado en el trabajo de fin de máster de Carlos Astúa, codirigido por Jonathan Crespo (autor de esta tesis) y Ramón Barber [7].

El análisis de contornos emplea la imagen de profundidad proporcionada por la kinect, que es el resultado de combinar la información del sensor de infrarrojos y la cámara RGB. La imagen resultante es representada en una escala de grises donde el valor de cada píxel representa la distancia a la que se encuentra. Cuanto más negro, más lejos está. Esto permite extraer el contorno de los píxeles que están a la misma distancia. Una vez que se extrae dicho contorno, la librería de OpenCV permite diferentes métodos de comparación, como el del área, redes neuronales y correlación, entre otros. Sin embargo, se ha tenido en cuenta que este sistema de detección de objetos iba a ser integrado en el navegador semántico, por lo que debería proporcionar una manera de generalizar la detección de objetos si se quiere que el robot pueda reconocer un objeto similar aunque no lo haya percibido antes. De todos los métodos contemplados, el único que ofrece mejores prestaciones en este sentido es el de la correlación, puesto que el área puede tener problemas en diferenciar objetos distintos pero con un área similar, las redes neuronales requieren el entrenamiento previo con cada objeto del entorno y otros métodos son demasiado específicos. La apuesta es poder reconocer, por ejemplo, una silla por su forma genérica y que el sistema reconozca cuándo está viendo una silla aunque esa silla en concreto no la haya visto anteriormente, simplemente por la forma genérica de dicho objeto. Las imágenes son tratadas como matrices de píxeles. Los valores contenidos en estas matrices son los que se emplean en la correlación, donde se compara el valor de cada píxel del modelo con el valor de los píxeles de la imagen y esto proporciona una matriz de correlación, que muestra cómo de similar es una matriz dada comparada con la imagen. En la

Figura 3.12: Detección de contornos.

Figura 3.12 se aprecia en la parte superior que hay dos sillas distintas pero cuyo nivel de correlación es alto, del mismo modo que se aprecia que con la imagen de la parte de abajo el sistema encontraría un nivel de correlación mucho menor.

En cuanto a los descriptores, hay que tener en cuenta que el procesamiento de imágenes es una tarea compleja debido a que los escenarios están constantemente cambiando. Esto no es simplemente al hecho de que se pueda añadir o eliminar objetos, ni a que los objetos puedan moverse, sino a que una imagen capturada en el mismo escenario sin cambios aparentes puede tener diferencias provocadas por factores como la iluminación o el ruido. Debido a este problema, encontrar similitudes entre dos imágenes del mismo escenario es uno de los grandes problemas de la visión artificial. La detección de características [64] se emplea cuidadosamente para resolver esto, se basa en extraer información que puede ser usada para identificar con precisión diferentes puntos de vista de una misma imagen. La detección de características trata de describir sólo aquellas partes de la imagen donde hay datos únicos o se pueden extraer descriptores. Para conseguir esto, primero hay que extraer los puntos clave de la imagen que son localizaciones bien distinguibles, como los bordes, marcas, etc. Esos puntos deberían ser muy repetibles. Entonces, las regiones vecinas se eligen alrededor de cada punto clave representado por un vector de características

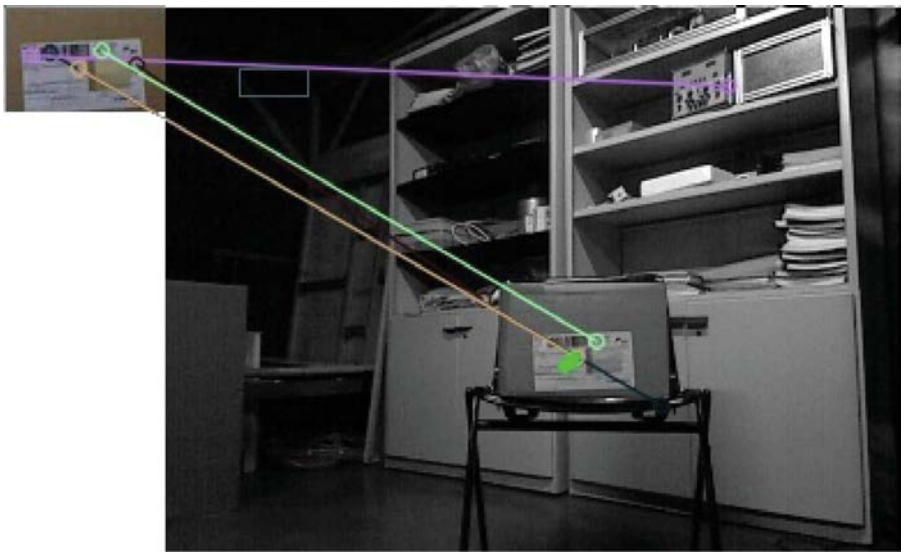


Figura 3.13: Detección de descriptores.

y unos descriptores de características únicos son procesados para cada región. Estos descriptores son fácilmente distinguibles además de robustos frente al ruido, a desplazamientos en la detección y a deformaciones geométricas y fotométricas. En este sistema de detección de objetos, se propuso un conjunto de detección de características para procesar descriptores fiables para la identificación de imágenes basado en el software SURF (Speeded Up Robust Features) [10], que es uno de los más empleados actualmente. La identificación de estos descriptores se basa en la distancia entre los vectores y esto es el principal factor a considerar cuando se está averiguando si la correlación encontrada es precisa. Para hacer esto, hay librerías como FLANN (Fast Library for Approximate Nearest Neighbors) [88] como ejemplo de una librería usada para resolver búsquedas rápidas del vecino más próximo. La Figura 3.13 muestra algunos puntos clave que conformaban los descriptores en pruebas realizadas durante el desarrollo de este método.

En este punto se han presentado dos técnicas diferentes. Una reconoce formas y permite una identificación más genérica de un objeto. La otra reconoce descriptores y permite una identificación específica de cada objeto. Ambas técnicas fueron fusionadas. Esta fusión implica que ambos algoritmos son utilizados para detectar el objeto. Esta aproximación crea un área común para ambos métodos para poder decidir si el

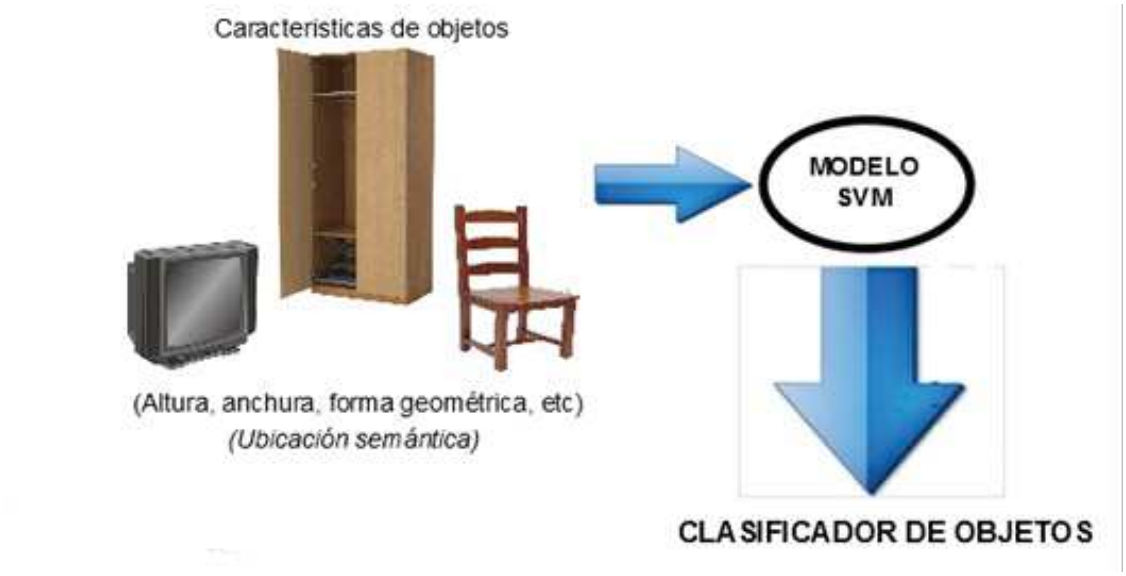


Figura 3.14: Detección de objetos empleando SVM.

objeto es el correcto o no. El método puede aplicarse usando datos provenientes del subsistema de contornos o del de descriptores, para la decisión final se emplean los resultados de ambos. Por ejemplo, si el contorno del objeto es un rectángulo y los descriptores muestran una caja, el resultado final será una caja. Pero si el contorno es triangular, entonces el resultado final probablemente no sea una caja. La ventaja de estos métodos es la habilidad de generalizar desde los contornos, dado que para cada contorno habría también un descriptor de imagen y los descriptores son muy específicos.

Detección de objetos por clasificación con SVM

El desarrollo de este método fue realizado en el trabajo de fin de máster de Alejandra Hernández, codirigido por Jonathan Crespo y Ramón Barber [58]. La idea es crear un clasificador basado en Máquinas de Soporte Vectorial para identificar objetos, como muestra la Figura 3.14.

El proceso de la detección de imágenes comienza preparando las imágenes para entrenar el sistema. Se toman varias imágenes de los objetos a entrenar y después son preprocesadas para facilitar la segmentación. Se obtiene una imagen con mejor

contraste y las áreas de interés quedan resaltadas. Después de esto comienza la segmentación. Se han empleado diferentes técnicas de segmentación para cada objeto que se pretende detectar. El siguiente punto es encontrar las características del objeto. En este sistema de detección, se han empleado dos alternativas para extraer las características de los objetos. Una es mediante descriptores de formas geométricas y la otra se basa en el modelo "bolsa de palabras" (Bag of words). Al completar este paso, se obtienen las matrices de entrenamiento y se definen los parámetros de una Máquina de Soporte Vectorial para construir el clasificador.

En la segunda etapa se toman imágenes en tiempo real con distintos formatos (imagen de color y de profundidad) mientras que se trata de reducir el ruido y eliminar píxeles inválidos. Después, ambas imágenes de color y profundidad pasan por una etapa de preprocesamiento similar a la fase de entrenamiento y ambas imágenes son segmentadas, lo que permite que se extraigan las características de cada objeto por separado. El objetivo final de usar diferentes tipos de técnicas de segmentación y extracción de características es evaluar la funcionalidad y comparar los resultados.

El paso final consiste en comparar la predicción del sistema con el objeto real. La Máquina de Soporte Vectorial (SVM) tras ser entrenada, clasifica el objeto. Finalmente se ejecuta la parte del proceso que localiza el objeto detectado en el entorno. Se calcula el centro de masas del objeto y la información de profundidad es usada para determinar la distancia y el ángulo del centro de masas respecto a la cámara. El sistema entonces envía toda la información relativa al objeto identificado y su posición en un mensaje de ROS publicado en un topic. El navegador semántico recoge este dato mediante el agregador de percepciones.

Este sistema de detección presenta resultados como los de la Figura 3.15, donde se muestra una clasificación correcta para el objeto *pantalla*. Los resultados muestran que el clasificador desarrollado presenta un porcentaje de verdaderos positivos de 60.94 % y una especificación igual a 70 %. La precisión del modelo es 65.47 %, con una tasa de fallos de clasificación de 34.53 %. Cuando el método de segmentación funciona correctamente y los objetos de la escena pueden ser separados mejor, la pantalla puede ser detectada correctamente. En [55] están publicados con detalle éste y otros resultados de este sistema de detección de objetos.

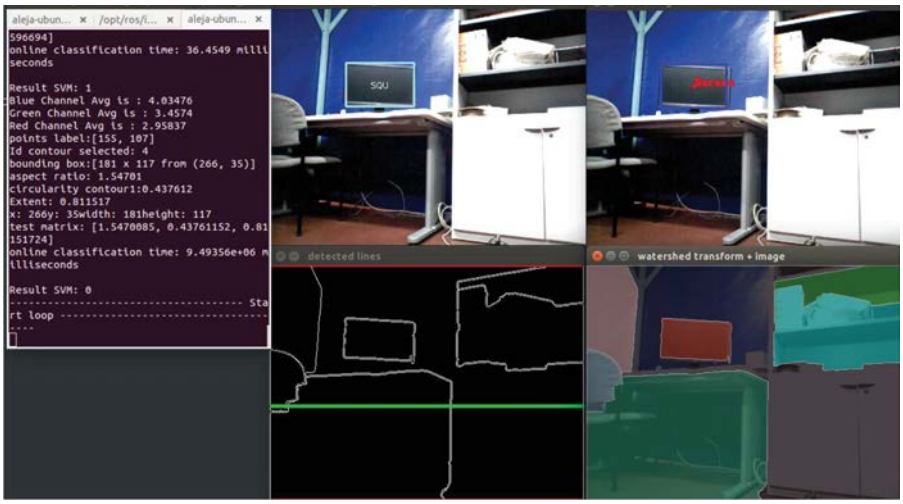


Figura 3.15: Detección de descriptores.

3.4.3. Detección de etiquetas

La dotación de la habilidad de detección de etiquetas, permitió en primera instancia que el robot tuviera un modo de identificar y reconocer objetos del entorno. El sistema basado en ARToolkit puede ser descargado e instalado libremente en cualquier máquina que trabaje con ROS. Para su funcionamiento no es necesario un sensor de profundidad como la kinect, con una cámara web sería suficiente. La estimación de la distancia entre la cámara y la etiqueta percibida la calcula sabiendo el tamaño de la etiqueta. Y mediante la deformación que produce la perspectiva en la imagen predefinida, el sistema calcula la posición relativa en el espacio de la etiqueta, así como su orientación. Las etiquetas son figuras impresas de marcos cuadrados con configuraciones diferentes en su interior, que se traducen en la pantalla por un cubo en la posición de la etiqueta como muestra la Figura 3.16.

El sistema funciona fluidamente sin necesidad de consumir grandes recursos. Esto dota al sistema de dos útiles habilidades: la detección de objetos etiquetados y la detección de marcas como puntos de refencia.

Empleo de detección de etiquetas como solución a la detección de objetos

El empleo de la detección de etiquetas permite detectar objetos que lleven puesta una etiqueta, la cual se asocia con dicho objeto en una base de datos. El sistema

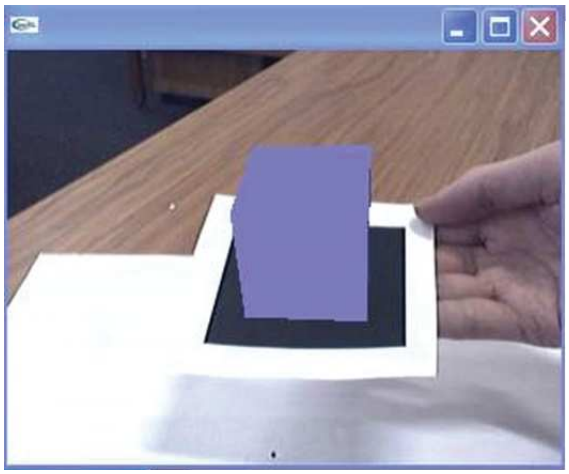


Figura 3.16: Etiqueta de realidad aumentada ARToolkit.

ha sido implementado de modo que si en el futuro se emplean nuevos detectores de etiquetas utilicen la misma base de datos. Soluciona por su simplicidad el primer problema que surgió con la intención de resolver la detección de objetos. Se pretendía obtener un sistema como el 1 de la Figura 3.17 pero debido a las dificultades iniciales comentadas anteriormente, se optó por el mecanismo 2 de la misma figura donde se perciben objetos etiquetados. Sin embargo este sistema no pretendió en ningún momento suplir permanentemente a un detector de objetos. Su objetivo se limitó a permitir hacer las primeras pruebas de detección y complementar la visión artificial del robot. Posteriormente los mecanismos 1 y 2 son utilizados simultáneamente por el módulo de percepción como se ve en la sección 3.4.1. De este modo los objetos bien entrenados por nuestros sistemas de clasificación se detectan tal cual, mientras que para objetos no entrenados o que presentan dificultades especiales para ser clasificados por sus características, se detectan añadiéndoles una etiqueta.

3.4.4. Mecanismos implementados para evitar fallos de detección

Cuando se incorpora la habilidad de la detección de objetos en un sistema, hay cierto tipo de errores frecuentes que hay que tratar de resolver. Y no simplemente hay que pensar cómo resolverlos, sino que también es importante decidir en qué subsistema

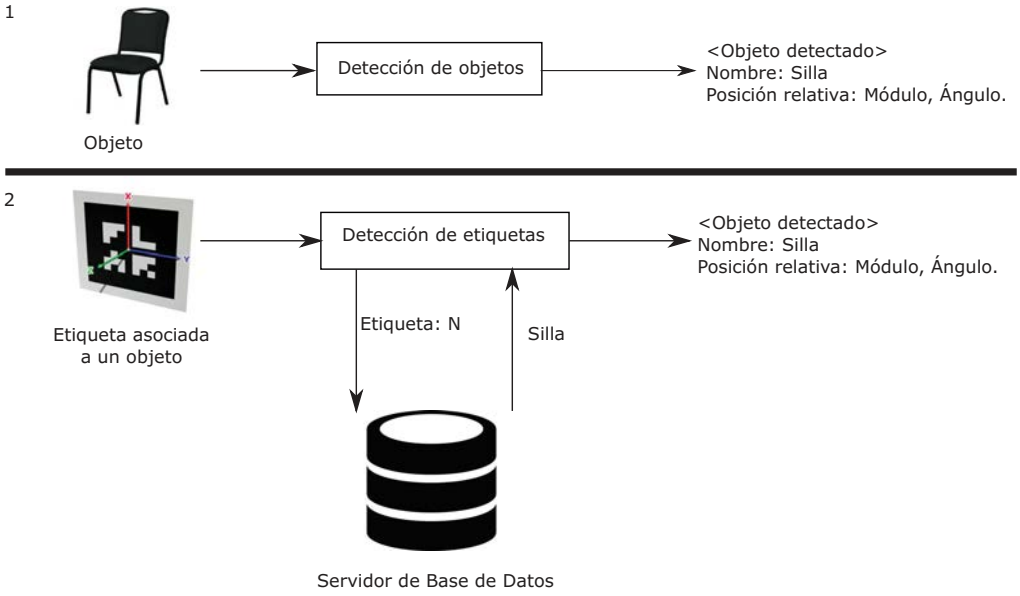


Figura 3.17: Empleo de un sistema de detección de etiquetas como un sistema de detección de objetos.

tienen que resolverse. Los fallos previsibles son de dos tipos: falsos negativos y falsos positivos. El primero es cuando hay un objeto y el sistema no lo percibe, el segundo es cuando no hay ningún objeto pero el sistema sí lo percibe.

El problema consistente en decidir qué parte del sistema debe ser el encargado de resolver los fallos previsibles es más delicado de lo que parece. Hay dos posibilidades, una es que la responsabilidad de filtrar los datos recae en el sistema sensorial. La otra es que dicha responsabilidad recae en los sistemas que reciben la información sensorial. La primera posibilidad hace que se envíen datos filtrados y depurados, la segunda hace que se envíen en bruto y que el que lo reciba lo filtre. La decisión tomada ha sido la de que sea el propio sistema sensorial el encargado de tratar de reducir los posibles falsos positivos. Respecto al problema de evitar los falsos negativos, se trata de resolver con comportamientos repetitivos del explorador y se espera que al final se haya detectado la gran mayoría de objetos. Si con una vuelta al entorno no se han percibido varios objetos, se estima que con unas cuantas vueltas más se percibirán la mayoría. Este problema no es excesivamente grave, pues se resuelve extendiendo el tiempo de las tareas del explorador.

En cuanto a los falsos negativos, en el agregador de percepciones se han tomado las medidas que se describen a continuación para tratar de reducirlos, dependiendo de lo que se esté detectando.

- Para la detección de objetos: Los falsos negativos surgían principalmente por errores de medidas en las posiciones de los objetos. Si, por ejemplo, hay una silla delante del robot, el sistema de detección de objetos emite constantemente que está viendo una silla con la posición de la silla relativa al robot. El problema venía porque el agregador de percepciones sólo puede diferenciar una muestra de otra por la distancia detectada. Siguiendo el ejemplo de la silla, si el agregador recibe que hay una silla a 1 metro de distancia y un instante después recibe que hay una silla a 2 metros de distancia y el robot no se ha movido, interpreta que son dos sillas diferentes. En cambio, si después de recibir que hay una silla a 1 metro de distancia, recibe que hay una silla a 1,1 metros de distancia, considera que se trata de la misma silla. Es por esto que es importante que el sistema de detección de objetos ofrezca una distancia fiable. Sin embargo, en la práctica, el detector de objetos es susceptible a múltiples factores que alteran constantemente las medidas de distancia de las muestras. Para solventar el problema se implementó el siguiente comportamiento en el agregador de percepciones: Se contabilizan las veces que se percibe el mismo objeto de forma consecutiva. Además, se guardan las distintas posiciones de cada detección. Cuando el mismo objeto ha sido detectado un número predeterminado de veces (este número se estableció en siete para las pruebas), las posiciones de los objetos detectados se ordenan en función de su distancia con el robot. Después de ser ordenados, se observa las medidas centrales, evitando de esta manera las muestras que tengan medidas muy dispares. En los experimentos, al estar establecido el número de muestras en siete, se tomaban las muestras tercera, cuarta y quinta (descartándose la primera, segunda, sexta y séptima). Y se comparan las distancias de esas tres muestras. Si la diferencia es mayor de otra cifra preestablecida de distancia máxima aceptable, todo el proceso se descarta. Si las medidas de distancia resultan ser similares, se acepta que realmente ese objeto debe de estar ahí y el agregador de percepciones publica en el topic de objetos detectados el nombre del objeto y la posición (tomando la mediana de las tres posiciones

elegidas). Además, la posición se da en coordenadas cartesianas siempre, por lo que si la posición que recibe el agregador de percepciones está en coordenadas polares, es transformada a cartesianas.

- Para la detección de etiquetas: El problema que produce falsos positivos en la detección de etiquetas es similar a la detección de objetos, con la diferencia de que la distancia no es tan relevante. Con las etiquetas detectadas con ARToolkit el problema es que el sensor a veces capta en su imagen un patrón similar a una etiqueta. Para minimizar este efecto sería más productivo mejorar la iluminación y trabajar en un ambiente más despejado. No obstante, el tratamiento de las distancias de medidas consecutivas que se ha visto en la detección de objetos también se aplica aquí, con la salvedad de que el nodo de ARToolkit publica un vector de etiquetas percibidas simultáneamente y por lo tanto hay que recorrer ese vector para saber qué etiquetas se están viendo en un momento determinado.

Además, se aplica otro filtro a la detección de objetos. En ocasiones, el sistema de clasificación de objetos utilizado emitía datos erróneos fácilmente detectables. Debido a que el sensor de visión del robot tiene unas conocidas limitaciones de campo visual, todas las detecciones de objetos en posiciones menores o mayores de lo que permite el rango de visión son descartadas. En concreto, la cámara Asus Xtion Pro utilizada presenta una distancia de uso entre 0.8 metros y 3.5 metros. Todos los objetos detectados a menos de 0.8 metros y más de 3.5 metros eran errores y por lo tanto se decidió descartarlos automáticamente, lo que redujo apreciablemente el problema de los falsos positivos.

3.5. Módulo interactivo

Se ha desarrollado un sistema de diálogo humano-robot, mediante el que el usuario puede realizar tres funciones principales:

- Solicitud de un destino. El usuario se comunica con el robot para solicitar un destino semántico. Este destino puede implicar localizar un objeto concreto, un objeto genérico, un objeto con alguna característica concreta, un lugar específico, un lugar genérico, un lugar donde se pueda realizar alguna acción concreta

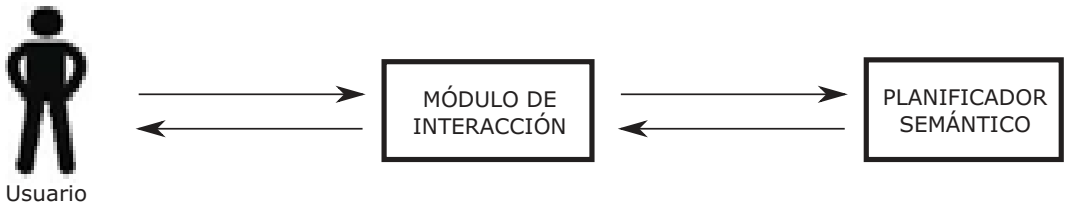


Figura 3.18: Módulo interactivo como mediador.

o un lugar donde se pueda realizar alguna acción con un efecto o significado concreto.

- Introducción de conocimiento en la base de datos del robot. Cuando el robot considera que no tiene suficiente información para decidir a qué lugar debe dirigirse, inicia un diálogo con el usuario mediante el cual puede adquirir nuevo conocimiento con el que deducir la ubicación del destino solicitado.

- Confirmación/negación de una consulta realizada por el robot. El usuario también deberá responder a las consultas que realice el propio robot, que excluyendo el aprendizaje, mayoritariamente serán motivadas para pedir la confirmación de que el destino alcanzado es el correcto.

La comunicación usuario-robot también permite que el robot pida la confirmación o la negación de una orden, así como de un destino. Una vez que el robot llega a un destino, pregunta al usuario si era un destino correcto o si, aún siendo un destino correcto, no es el destino exacto que buscaba el usuario.

3.5.1. Descripción del módulo interactivo

En la Figura 3.18 se muestra el flujo de información de este módulo, es el mediador entre el usuario y el sistema. Los casos de uso posibles en la petición de destino por parte del usuario se recogen en la Figura 3.19. De esta última figura, se puede deducir el tipo de información que requiere el sistema para la situación de solicitar un destino. Es la primera necesidad del sistema que implica desarrollar una interfaz de usuario: llegar al entendimiento del lugar al que el usuario quiere llegar.

El navegador semántico desarrollado a lo largo de esta tesis parte de la base de que las peticiones humanas de desplazamiento están motivadas para satisfacer una necesidad o cumplir un objetivo concreto. Se asume además que el ser humano es una máquina clasificadora, con lo que tiende a guardar en los mismos lugares los objetos que sirven para realizar las mismas tareas o tareas relacionadas. Se contempla además la posibilidad de que ciertas tareas tengan un significado asociado, como por ejemplo *jugar es divertido*. Así pues se ha diseñado la interfaz de usuario suponiendo que las peticiones para llegar a un destino estarán dentro de uno de estos casos:

- Objeto. Esto es cuando el usuario está buscando un tipo de objeto de forma genérica. Por ejemplo: refresco, agua, silla, etc. No manifiesta lo que quiere hacer, directamente quiere encontrar un tipo de objeto determinado.

- Objeto con característica. La interfaz de usuario debe poder ofrecer la oportunidad de que se especifique una característica, que podría cambiar completamente el lugar donde buscar. Por ejemplo: agua fría, ropa limpia, armario de cocina, etc.

- Objeto en un lugar. Si el usuario concreta un lugar, la búsqueda se ciñe a ese lugar. Ejemplo: la televisión del salón.

- Acción. El usuario puede expresar el deseo de realizar una acción, sin especificar el objeto que le puede servir. Por ejemplo: imprimir un documento.

- Estancia. Es el caso en el que el usuario solicita ir a una estancia concreta. Por ejemplo: ir a la cocina.

- Característica. Se contempla también el caso de que el usuario tan sólo comunique la característica de lo que desea, sin mencionar el objeto. Por ejemplo, si le apetece beber algo caliente.

- Significado. Este último término puede ser un poco confuso por la ambigüedad de la palabra. Lo que se pretende captar en este caso es cuando el usuario hace una petición de ir a un lugar inespecífico pero que tiene una característica

concreta, etiquetada por un significado extra con la que el usuario haya identificado una acción. Por ejemplo, si el usuario asocia la acción *descansar* con algo *tranquilo*, tal vez su petición al sistema sea únicamente *ir a un sitio tranquilo*.

Para recoger todas las situaciones en las que el usuario puede manifestar una petición de destino, se ha definido un tipo de mensaje que contiene un campo para cada unidad de información que pudiera dar el usuario. El módulo de interacción tiene que rellenar estos campos, de manera que así se puede deducir posteriormente el deseo del usuario. El módulo de razonamiento es el responsable de interpretar la información implícita del hecho de tener ciertos huecos rellenos y otros no, y de traducir el destino que solicitó el usuario a un destino geométrico o topológico que de esta manera llevan al robot a un lugar físico concreto del entorno, permitiendo la navegación.

Con el sistema recién iniciado, el módulo de interacción está en un estado de esperar a recibir una petición por parte del usuario. Bien sea mediante teclado o mediante voz, cuando el usuario comunica que quiere alcanzar un destino, el módulo de interacción rellena los huecos de información anteriormente comentados. Cuando extrae todos los datos que aporta el usuario, se forma un mensaje del tipo *Destino*. Este mensaje es la recopilación de todos los huecos de información y se envía al topic *solicitudDestino*, al cual está suscrito el gestor del navegador semántico.

El mensaje finalmente lo recibe el módulo de razonamiento, que en este caso tiene como función analizar el mensaje recibido y deducir a qué lugar del entorno debe dirigirse. Cuando el robot llega a la posición requerida, el módulo de razonamiento recibe esa notificación y procede a mandar otra petición al módulo interactivo. Esta nueva petición se realiza con el objeto de pedir una confirmación al usuario sobre el destino alcanzado. Los procesos de razonamiento del robot son transparentes al usuario, así pues si hay algo erróneo en la base de conocimiento, si el sistema ha malinterpretado la información del usuario o ha habido algún otro fallo de distinta naturaleza en el proceso de inferencia, el usuario no sería consciente del motivo del fallo. Esta es una de las razones de pedir confirmación del destino, pues el usuario recibe información de lo que trataba de alcanzar el robot. Además, la respuesta es procesada por el sistema, de tal manera que en futuras búsquedas da prioridad a los lugares que acumulan mayor número de confirmaciones exitosas. Por lo tanto, el módulo interactivo recibe un mensaje que le ordena pedir confirmación al usuario. Esta confirmación será obtenida

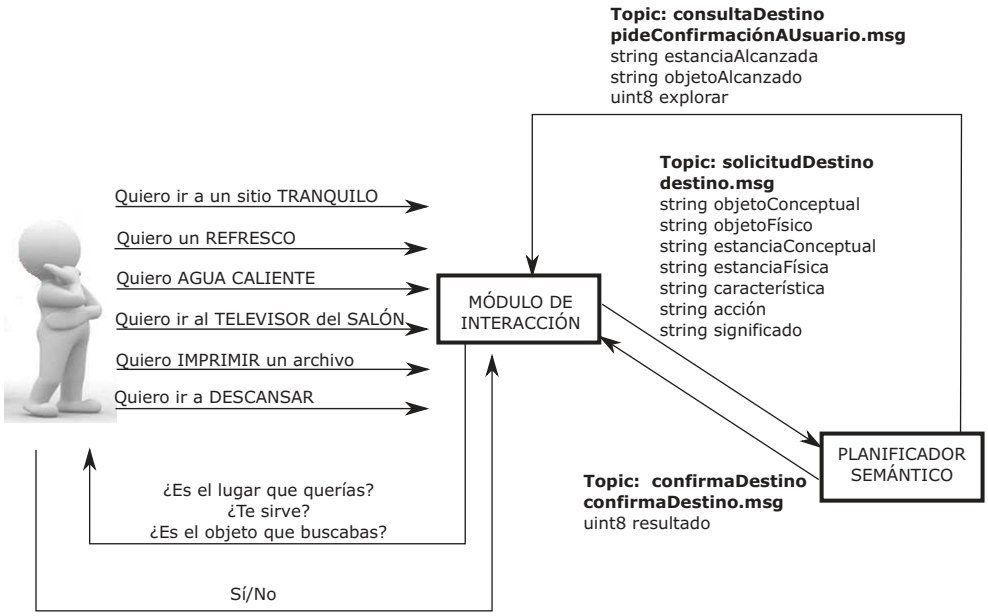


Figura 3.19: Módulo interactivo como mediador.

mediante una respuesta que será interpretada como *SÍ* o como *NO*. En cualquier caso, la respuesta vuelve a ser enviada mediante un mensaje que transmite el módulo de interacción y que recibe el módulo de razonamiento.

La comunicación entre el módulo de interacción y el gestor del navegador semántico debe ser mantenida y de este modo es sencillo sustituir los módulos por otras implementaciones. Aunque la primera aproximación se implementó con una interfaz de teclado, posteriormente se ha añadido una implementación orientada a diálogo con voz.

El sistema de comunicación con el usuario ha sido diseñado teniendo en cuenta los módulos del esquema de la Figura 3.20, donde se ve también la información relevante intercambiada entre estos módulos mediante topics o servicios. En esta figura se contempla cuando el usuario hace una petición de acciones con un significado concreto, como por ejemplo, "quiero hacer algo *divertido*". El resto de características del navegador han sido omitidas en esta figura para enfocar la atención únicamente en la interacción y el desplazamiento del robot. Los módulos corresponden a las elipses y son los siguientes:

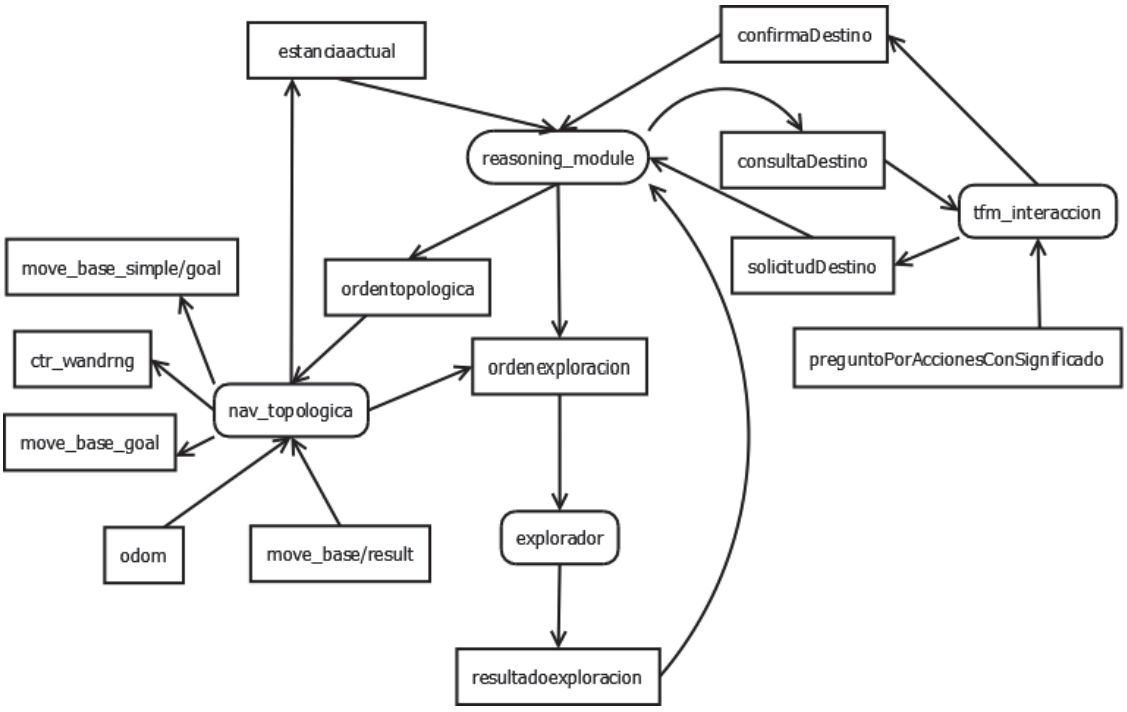


Figura 3.20: Esquema del navegador y el módulo de interacción.

- reasoning_module: Este módulo es que corresponde al motor de razonamiento que está detrás de todo el sistema de navegación semántica, actuando muchas veces como el centro de toda la arquitectura. En lo concerniente a la interacción con un usuario, se destaca la recepción de tres datos: la estancia actual, el estado de exploración y la solicitud del destino. De estos, tan sólo el último es el que viene directamente del usuario. Los otros dos provienen del navegador de bajo nivel (en la figura corresponde con un navegador topológico) y del explorador, que aportan información de la estancia actual donde se encuentra el robot y del resultado de una tarea de exploración, respectivamente.

- nav_topologica: Es el módulo correspondiente al navegador de bajo nivel (considerando que la navegación a alto nivel es el semántico). Si bien la evolución del mismo ha sido llegar a una navegación topológica, para el estudio y desarrollo del módulo de interacción con el usuario se ha hecho que este módulo utilice un navegador geométrico. Este hecho es transparente para la interacción, pero se refleja en la figura por el envío de información a los topics move_base_simple/goal y move_base/goal (para enviar la coordenada destino) y la recepción de move_base/result, que notifica la llegada a dicha coordenada de destino.

- tfm_interaccion: Es el módulo que gestiona la interacción con el usuario. La primera versión de este módulo se implementó como una interfaz por teclado, posteriormente dió lugar al tutelaje de un Trabajo Fin de Máster centrado en desarrollar un diálogo lo más natural posible que cubriera esta necesidad de interacción con el robot [84].

3.5.2. Integración del módulo interactivo en el sistema

Independientemente de la implementación realizada, el módulo de interacción debe cumplir con la interfaz pertinente para funcionar en el sistema. Esta interfaz no hay que entenderla desde el punto de vista de la implementación en código C++, puesto que las llamadas entre módulos no se realizan mediante llamadas a funciones, sino mediante el intercambio de mensajes en topics dentro de un entorno de ROS. De este modo, la implementación concreta del módulo es muy abierta y sólo exige que

se dé soporte a unos servicios establecidos, se transmita información en unos topics concretos y se obtenga información de otros topics del sistema.

Servicios atendidos

Un servicio en ROS es una petición en forma de mensaje que espera una respuesta. El módulo interactivo atiende los siguientes servicios:

- Petición de acciones con un significado. Este servicio pretende recopilar información sobre las acciones que suscitan un sentimiento subjetivo en el usuario. El proceso del servicio inicia un diálogo con el usuario para conseguir esa información. Por ejemplo, para que el usuario diga qué acciones considera que son divertidas.

- Pregunta de objetos por utilidad. Este servicio hace que el sistema pregunte al usuario por los objetos que sirven para una acción, por ejemplo objetos que sirven para descansar.

- Pregunta por interacción. Este servicio inicia un diálogo con el usuario para asociar interacciones entre objetos. Por ejemplo, podría hacer que el robot preguntase con qué otros objetos del entorno se relaciona un libro. El usuario podría responder que con la estantería (porque está sobre ella).

- Pregunta de confirmación. Este servicio se emplea siempre y cuando se necesite una confirmación por parte del usuario.

Servicios como cliente

- Petición de objetos por acción. En este caso el sistema de interacción con el usuario es un cliente porque solicita la información al navegador. Esta petición se resuelve con el conocimiento del navegador semántico sobre los objetos y sus utilidades. Se emplea cuando el sistema de diálogo debe enumerar al usuario los objetos y sus utilidades que son conocidos por el sistema (es decir, que están en la base de datos del sistema).

Suscripciones a topics

- Consulta de destino. Este topic publica la necesidad de confirmar el destino final del robot.

Publicaciones en topics

- Confirmación de destino. En este topic se publica la respuesta del usuario sobre la confirmación del destino final del robot.
- Solicitud de destino. Esta publicación contiene el destino semántico hacia el que se quiere dirigir el usuario.

3.6. Módulo del navegador de bajo nivel

El módulo del navegador de bajo nivel es en realidad una interfaz para que el sistema de navegación semántica se conecte con un sistema de navegación geométrico o topológico. Es el módulo encargado de gestionar el navegador sobre el que se asienta la capa semántica. Si se toma como premisa que el navegador de bajo nivel (así se llaman en esta tesis a los navegadores geométricos o topológicos) está implementado en ROS, este módulo puede ser implementado de manera genérica para ambos tipos de navegadores. Lo que se tiene que tener en cuenta es llevar el control de los topics que usa el navegador para recibir peticiones de destino, publicar la posición actual y publicar también el resultado de la petición.

3.6.1. Descripción del módulo integrador de la navegación de bajo nivel

El flujo de topics y relaciones de este módulo con el resto del sistema queda representado en la Figura 3.21. El navegador de bajo nivel se comunica con el planificador y el explorador semántico mediante los siguientes topics:

- <estanciaActual>. Puesto que el navegador de bajo nivel es el que realmente mueve el robot por el entorno, despreocupándose de la jerarquía conceptual,

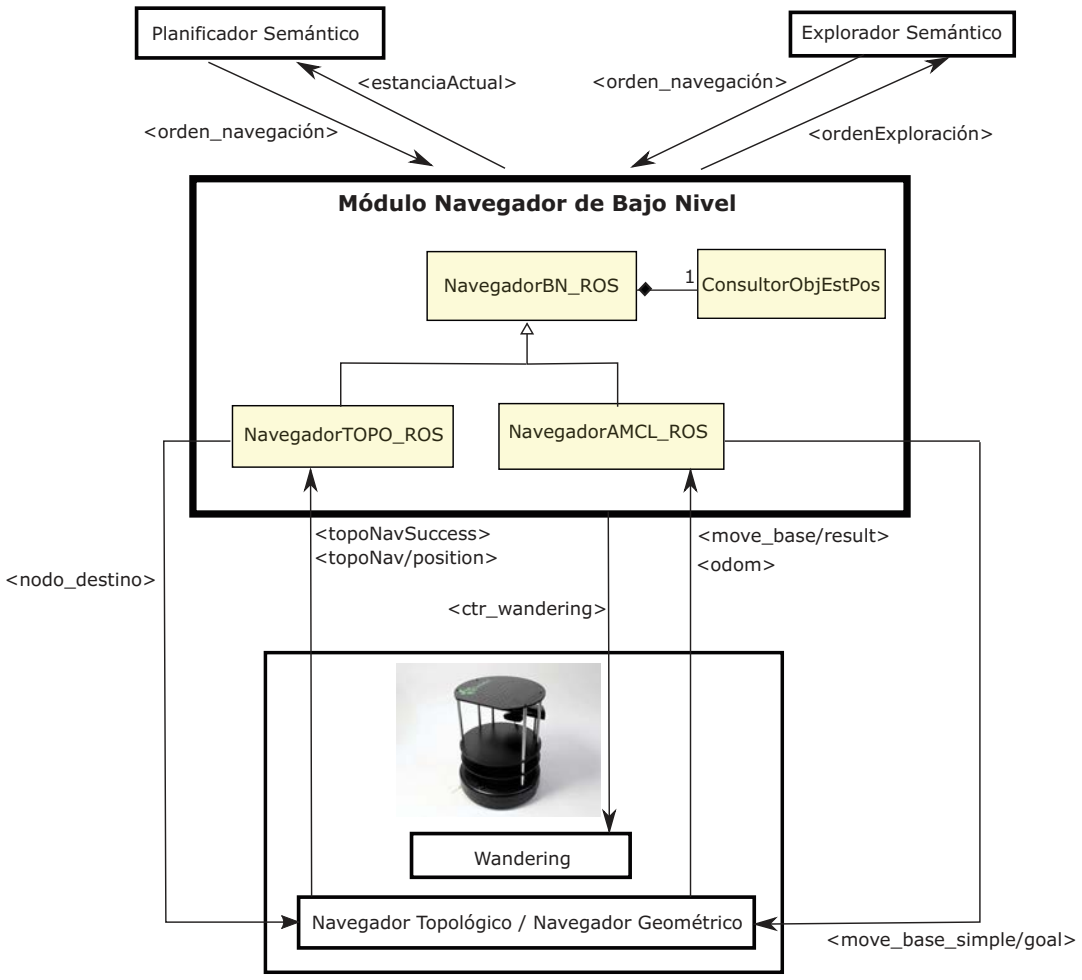


Figura 3.21: Módulo interfaz con el navegador de bajo nivel en el sistema.

es el encargado de publicar la posición del robot dentro de la jerarquía espacial. El topic "estanciaActual"se publica un mensaje de tipo *PosiciónBN*, que contempla de manera genérica la posición entendida como nodo o coordenada.

- <orden_exploración>. El navegador de bajo nivel también requiere en ocasiones comunicarse con el explorador semántico. Cuando este módulo recibe la confirmación de haber alcanzado una posición pedida, se le solicita al explorador que pase al estado NO_EXPLORANDO.

- <orden_navegación>. A través de este topic, el navegador de bajo nivel recibe órdenes por parte del explorador o del planificador semántico. Del primero puede recibir la orden de moverse erráticamente por un lugar determinado (wandering). Del segundo recibe peticiones de desplazamiento hasta un objeto o posición determinada. También puede recibir la orden de moverse a un punto relativo al robot, por ejemplo, si se tiene que desplazar a 2 metros hacia delante y 1 metro a su derecha (esto podría ser el resultado de tener que aproximarse a un objeto que está en su campo de visión). Para cumplir esto es obligatorio que exista en el sistema un navegador geométrico, independientemente de que pueda existir otro navegador de bajo nivel como uno topológico.

- <ctr_wandering>. Este módulo activa o desactiva la habilidad wandering que está en el robot.

- <move_base_simple/goal>. Este es el topic donde el navegador basado en AMCL de ROS recibe las peticiones de destino como coordenadas relativas a un mapa geométrico.

- <move_base_result>. Este es topic donde el navegador basado en AMCL de ROS publica el resultado de su navegación. Por este topic el sistema se entera de si el robot alcanzó la posición deseada o no.

- <odom>. Este topic publica la odometría del robot, con la cual se puede calcular su posición geométrica en un mapa.

- <nodo_destino>. En este topic el sistema publica el identificador del nodo que se pretende alcanzar. Es empleado cuando el navegador de bajo nivel que se utiliza es el navegador topológico.

- <topoNavSuccess>. En este topic el navegador topológico publica el resultado de su navegación.

- <topoNav/position>. En este topic se publica la posición actual del robot según el navegador topoógico. Corresponde al identificador del nodo en el que se encuentra.

En la misma Figura 3.21 se observa que el navegador de bajo nivel también tiene un objeto de la clase ConsultorObjEstPos. Esta clase agrupa una serie de métodos destinados a la obtención de las posiciones de los objetos y de las estancias y se puede utilizar independientemente de que el navegador utilizado sea topológico o geométrico, puesto que guarda la posición en forma de coordenadas o de identificador de nodo mediante un objeto de la clase PosiciónBN (Posición de Bajo Nivel). La Figura 3.22 muestra con más detalle los atributos y métodos de las clases implicadas (se obvia aportar esta información de la clase PosiciónBN por su simplicidad, puesto que consiste únicamente en atributos para almacenar la posición geométrica y el identificador de un nodo, y de los métodos para acceder a ellos). El constructor de ConsultorObjEstPos (Consultor de Objetos, Estancias y sus Posiciones) tiene como argumento una variable que inicia el objeto en modo geométrico o topológico. En función de este modo, se utiliza uno de los dos clientes de servicios ROS que responden con posiciones o nodos a las peticiones sobre objetos y estancias. Además en la última figura se muestran las clases heredadas de NavegadorBN_ROS. En el siguiente apartado se comenta la utilidad de esta relación y cómo se aprovecha para integrar los distintos tipos de navegadores.

3.6.2. Integración con distintos navegadores: interfaz de navegador de bajo nivel

En el trabajo realizado en esta tesis, siempre se ha tenido presente la idea de que la navegación semántica puede mejorar cualquier sistema de navegación convencional.

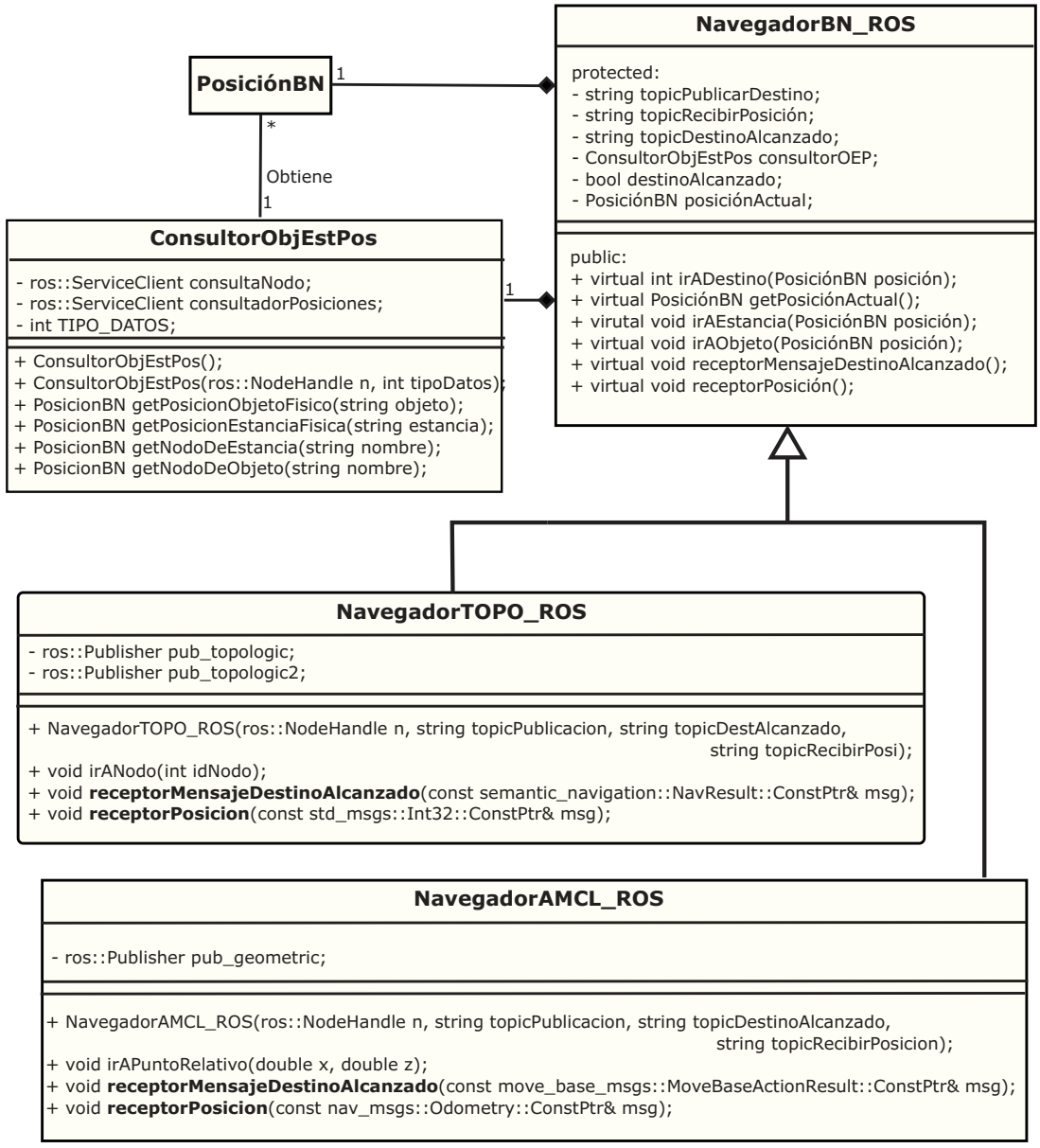


Figura 3.22: Clases que heredan de NavegadorBN_ROS.

La navegación semántica puede ser una capa que se construye sobre un navegador con menor nivel de abstracción. Por ello, desde las primeras fases de diseño se ha tenido en cuenta que este navegador semántico debe poder ser utilizado por cualquier otro navegador de forma inmediata y casi transparente. Esto es un objetivo muy ambicioso, y por ello ha sido acotado a navegadores de menor nivel implementados en ROS. Y se han contemplado los dos tipos de navegadores más utilizados hasta el momento que no incorporan semántica en su filosofía: los navegadores geométricos y los navegadores topológicos.

Por ello, se ha creado la clase **NavegadorBN_ROS** (Navegador de Bajo Nivel implementado en ROS). La idea es que cualquier navegador hecho en ROS funciona interaccionando mediante los siguientes topics:

- Un topic para recibir la petición de destino. Todo navegador necesita que alguien le diga a dónde tiene que ir. Los navegadores geométricos reciben esta información como coordenadas de un mapa conocido. Por ejemplo, ir al punto (3,5) del mapa. Los navegadores topológicos lo que necesitan es saber el identificador del nodo al que tienen que ir. Por ejemplo, ir al nodo 3. La clase NavegadorBN_ROS tiene el atributo *topicPublicarDestino*, que es donde se guarda el nombre del topic que utiliza el navegador de bajo nivel para recibir dicho destino. El navegador geométrico utilizado, que está disponible en los paquetes públicos del Turtlebot y está basado en el algoritmo de AMCL, hace que el robot se desplace hacia la coordenada publicada en el topic *move_base_simple/goal* mientras que el navegador topológico utilizado recibe las peticiones que se publican en *nodo_destino*.

- Un topic donde el navegador de bajo nivel publica el resultado de su navegación. Cuando el navegador considera que ha alcanzado su destino, publica esta información en un topic. Igualmente si hubo algún error también es publicado. El navegador geométrico utilizado publica en el topic *move_base/result*, mientras que el navegador topológico que se ha integrado publica en el topic *topoNavSuccess*.

- Un topic para transmitir la posición actual del robot en el propio sistema de navegación. Todo navegador debe trabajar empleando la posición actual, este es

un dato que debe conocer y transmitir. En el caso de la navegación geométrica en realidad el topic donde se encuentra esta información se publica en *odom* y se trata de la odometría. En cambio, en el navegador topológico, el sistema publica el nodo actual donde se encuentra el robot en el topic *topo_nav/position*.

Considerando que estos topics son la base de la integración de cualquier navegador con el sistema de navegación de esta tesis, forman parte de la superclase NavegadorBN_ROS y las dos subclases correspondientes a los dos tipos de navegación de menor nivel considerados, heredan estos atributos. Esta estructura de clases se muestra en la Figura 3.22. Otros atributos necesarios para el correcto funcionamiento del sistema y que deben pertenecer a la superclase son un objeto de la clase ConsultorObjEstPos, del cual se utilizan sus métodos para obtener las posiciones de objetos y estancias; una variable booleana para saber cuándo el navegador ha llegado al destino y un objeto de la clase PosiciónBN para tener actualizada la posición del robot.

Puesto que se va a tratar con navegadores geométricos y topológicos, se han creado dos subclases correspondientes a cada tipo de navegación. Comparten todo lo heredado de la superclase, únicamente difieren en la implementación de sus métodos, como es natural; en la inclusión de métodos propios (*irANodo* para el caso del navegador topológico e *irAPuntoRelativo* para el navegador geométrico) y en las cabeceras de los métodos de subscripción a topics *receptorMensajeDestinoAlcanzado* y *receptorPosicion*. La razón de esto último es que no basta con conocer el nombre del topic para subscribirse a él, también hay que modificar el tipo del mensaje recibido.

La integración de diferentes navegadores se ha tratado de hacer lo más genérica y sencilla posible. Así pues, en el módulo del *Integrador del Navegador de Bajo Nivel* se sigue el esquema de la Figura 3.23. Lo que se hace es instanciar un objeto de una subclase de NavegadorBN_ROS acorde al modo en el que se lanza el módulo. Para ello se instancia un objeto de NavegadorAMCL_ROS para la navegación geométrica y un objeto de NavegadorTOPO_ROS para la topológica, indicando en sus constructores los tres topics de los que se ha hablado en esta sección. El siguiente paso es asignar a la variable global *NavegadorBN* la instancia recién creada. Puesto que son subclases de la misma superclase, no hay ningún problema y a partir de ese momento en *NavegadorBN* se encuentra el navegador seleccionado. El módulo no necesita saber cuál es puesto que las funciones son llamadas desde este objeto empleando el polimorfismo

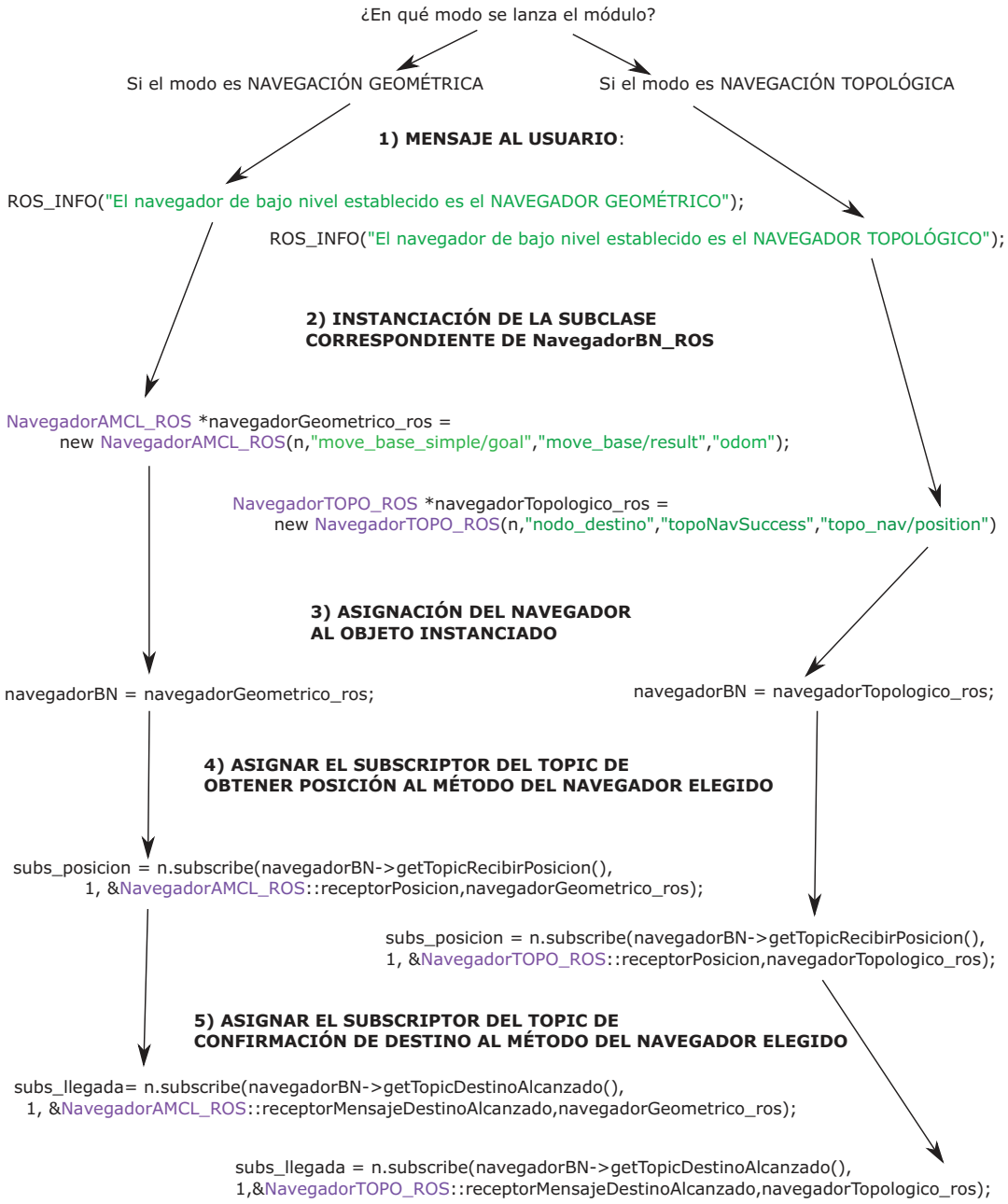


Figura 3.23: Esquema del proceso de instanciación del navegador de bajo nivel.

de clases. Lo único que hay que asignar aún son las funciones de atención a los topics, es decir, las funciones mediante las que se subscriben los topics. De los tres topics mencionados, dos son para recibir información. Uno para la posición actual que publique el navegador y otro para el resultado de la navegación. Una vez asignados los subscritores, el módulo está listo para funcionar de igual manera independientemente del navegador de menor nivel utilizado.

3.6.3. Estados del navegador de bajo nivel

El funcionamiento de este módulo requiere de un sencillo diagrama de estados que se muestra en la Figura 3.24. Básicamente es para gestionar las distintas peticiones y conocer si el robot en ese momento se está desplazando o no. Los eventos que provocan cambios de estado son de dos tipos: Por petición de ir a un destino o por recepción del resultado de una navegación.

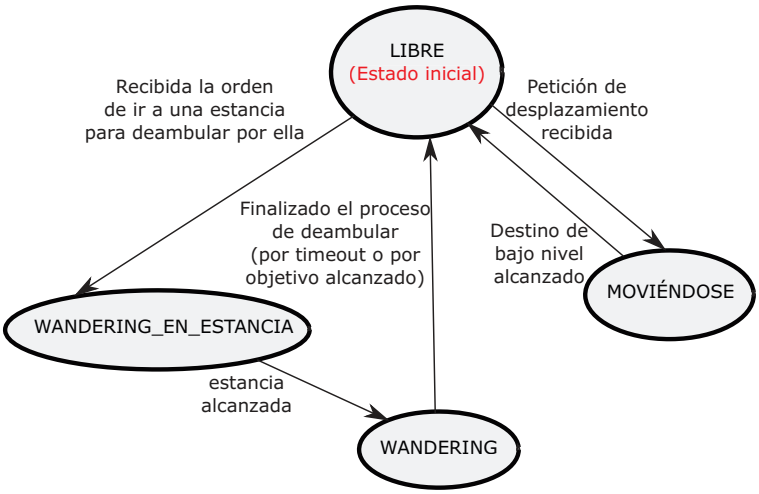


Figura 3.24: Diagrama de estados del módulo navegador de bajo nivel.

Los estados del módulo navegador son:

- Libre. Es el estado inicial, indica que el módulo no ha recibido ninguna petición de desplazamiento.

- Moviéndose. Este estado indica que el módulo ha recibido una petición de desplazamiento y el robot se encuentra en ruta hacia algún lugar concreto. El destino es conocido y el módulo sabe a dónde va.

- Wandering_en_estancia. Este estado indica que el módulo ha recibido una petición de deambular en una estancia determinada. El explorador semántico es el que normalmente realizará peticiones de esta naturaleza. Aquí el robot se dirige a una estancia concreta pero una vez allí no tendrá destino conocido.

- Wandering. Este estado normalmente es alcanzado después de haber llegado a una estancia donde el explorador requiere entrar en modo wandering. El módulo no utilizará el navegador de bajo nivel para esta tarea hasta que nadie lo indique, la función wandering la realiza un nodo obtenido de un repositorio público.

Esto concluye la presentación de la arquitectura del sistema. Han quedado recogidos todos los módulos que constituyen el navegador. También se han explicado las interacciones entre ellos y el funcionamiento particular de cada uno.

CAPÍTULO 4

Gestión de la información semántica

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica." — Aristóteles

La navegación semántica puede ser definida como un sistema de navegación para robots móviles que tiene en cuenta las relaciones entre conceptos, sobre todo conceptos relativos a objetos y estancias. La información proporcionada por estas relaciones puede ser empleada en tareas como la localización del robot, el mapeo, la exploración y la planificación de trayectorias. En [135] se pueden encontrar detalles sobre las técnicas de navegación existentes. En el caso de la navegación semántica es necesario gestionar la información de los conceptos que definen un entorno. Esto hace que normalmente la navegación semántica se apoye en una ontología [95], para manejar las relaciones entre conceptos y las jerarquías que se establecen entre ellos. En esta ontología se representa el modelado del entorno.

4.1. Modelado del entorno

El modelado del entorno es una abstracción que permite representar las características del entorno que son necesarias para realizar la tarea de navegación. Por

lo tanto, el modelado asienta las bases sobre las que se debe construir e interpretar el mapa del entorno que usa el robot. La complejidad del modelado es variable, dependiendo de los elementos del entorno que hayan sido considerados relevantes por aportar información útil para la navegación o localización. En un navegador puramente geométrico, el modelado del entorno se puede reducir a representar el espacio libre y diferenciarlo del espacio ocupado. Por lo tanto, una idea comúnmente empleada es representar los obstáculos como polígonos según su contorno. Según ese modelado, el mapa que necesitase un navegador de este tipo se reduciría a posicionar de manera absoluta los obstáculos en un mapa geométrico en el que cada punto del mapa se corresponde con una coordenada del mundo real. Sin embargo, un robot de servicios que trabaje en un entorno humano desaprovecha la información que aporta el entorno si reduce su modelado a un mapeo geométrico. Los robots provistos de modelados semánticos de los lugares donde se mueven, tienen mayor capacidad de decisión autónoma, son más robustos y eficientes.

La información que emplea un navegador semántico requiere mayor nivel de abstracción. Lo que se quiere conseguir con estos navegadores es que el robot pueda entender órdenes relativas a lugares del entorno que no están asociados de forma específica con un punto espacial. Son órdenes concretas de forma conceptual. Esto quiere decir que no hace falta tener un conocimiento de la coordenada exacta de dónde está el objetivo a alcanzar, siendo suficiente con precisar el concepto del objeto o lugar de destino. Además el sistema debe disponer de gran autonomía para saber hacia dónde tiene que dirigirse para cumplir cierto objetivo. Para entenderlo mejor, supongamos que el robot recibe la orden de ir a un sitio *divertido*. Lo que haría un oyente humano joven en esta situación, podría ser ir al salón porque allí está la consola. Para que el robot deduzca que tiene que ir al salón porque hay una consola que sirve para jugar y jugar es divertido, necesita que el modelado del entorno contemple un tipo de información específica, que son una serie de relaciones entre conceptos. Estas relaciones son principalmente entre conceptos sobre objetos y sobre el lugar donde se encuentran. También son relevantes las relaciones entre objetos y otros objetos sin olvidar las relaciones entre objetos y su utilidad. En los escenarios estudiados, el entorno en el que se mueven las personas está rodeado de objetos creados o modificados para un propósito. Todo lo que no sean artefactos no se tiene en cuenta por el navegador

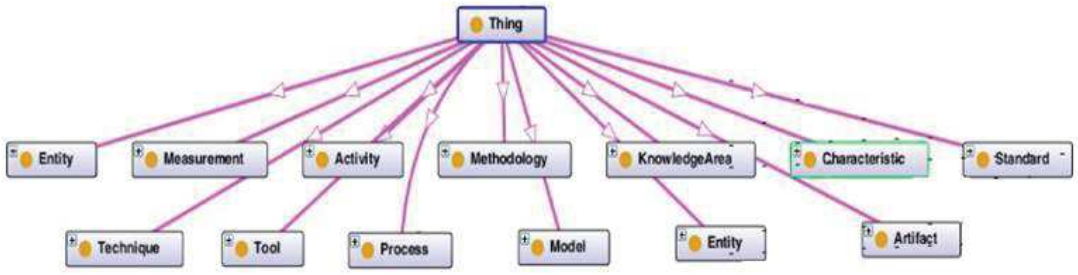


Figura 4.1: Representación de relaciones ontológicas.

semántico. Por lo tanto, los objetos del entorno tienen una utilidad, un propósito por el que han sido construidos o modificados. Esta utilidad también se puede relacionar con otros conceptos más subjetivos y personales para un usuario. En el ejemplo anterior se puede ver esta idea si relacionamos *jugar* con *divertido*.

Para abordar el problema, el robot necesita el conocimiento de las relaciones que hay entre los conceptos y el modelado debe incluir alguna manera de gestionar esta información. En el caso de la navegación semántica, una gran cantidad de información relativa al entorno y los objetos que contienen, necesita ser extraída y representada. Es por eso por lo que este tipo de navegadores suelen apoyarse en ontologías [102] para establecer las relaciones entre conceptos y conectarlos con objetos y lugares del mundo real. Los llamados *mapas semánticos* ya han sido utilizados para mejorar planificadores [37]. El modelado en estos navegadores trata de unificar la información de una ontología como la de la figura 4.1 con la información espacial de un mapeo topológico como el de la figura 4.2 o geométrico.

Muchos trabajos se apoyan en motores de razonamiento donde se incluyen la definición de reglas y una lista de hechos. En [43] usan el sistema de representación de conocimiento LOOM. En [44] emplean NeoClassic. Estos lenguajes están basados en lógicas descriptivas. Las cuales tienen un conjunto de reglas y una lista de hechos. Las inferencias se realizan correlacionando la lista de hechos con las reglas para comprobar si alguna regla encaja con algún hecho, lo que permite añadir un nuevo hecho (este proceso es una inferencia, puesto que se deducen nuevos hechos). El conjunto de reglas cambia de un entorno a otro cuando se usan motores de razonamiento. Con el sistema

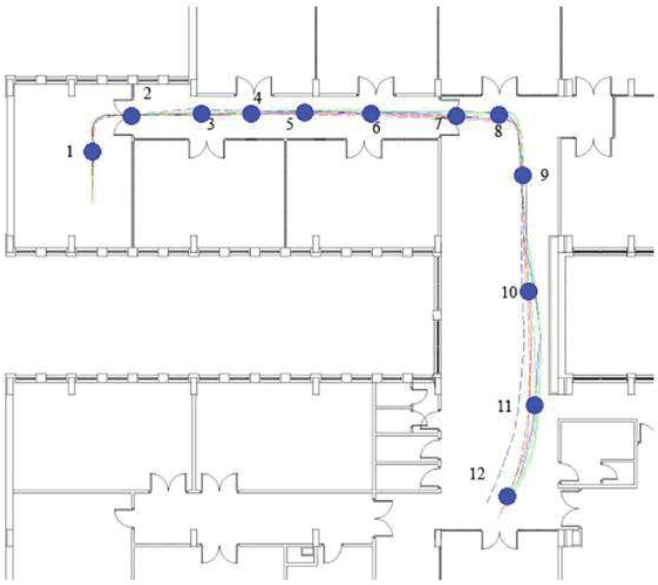


Figura 4.2: Mapeo topológico.

presentado en esta tesis sólo cambian los datos de la información del entorno (no las reglas).

En la sección 4.1.1 se propone un modelo relacional como alternativa al empleo de un motor de razonamiento convencional. En cualquier modelo relacional, las relaciones entre objetos pueden conectarse fácilmente con el grafo del nivel topológico del sistema de navegación o ser asociadas a una coordenada geométrica. El sistema propuesto también almacena la información percibida en tablas y al mismo tiempo la gestiona con un sistema de inferencia que accede directamente a una base de datos. El empleo de este sistema de razonamiento ha sido sometido a numerosas revisiones antes de decidir su inclusión en el sistema de navegación propuesto. Ha sido incluído tras comprobar que las pruebas de funcionamiento fueron satisfactorias, tal como se muestra en el capítulo 6 y además aportó algunas ventajas que no eran permitidas por un motor de razonamiento convencional. Este sistema fue presentado por primera vez en la conferencia de [30].

Una de estas ventajas de la implementación realizada es la capacidad de poder cambiar el entorno sin alterar el modelo relacional. Si se emplea un motor de razonamiento, la lista de reglas deben ser cambiadas por un programador o por una

Inteligencia Artificial avanzada que pueda hacerlo cada vez que se cambie el tipo de entorno. Esto es porque los tipos de estancia no se definen igual si estamos en una vivienda, oficina, hospital, etc. En un motor de razonamiento, la definición de un tipo de habitación implica añadir la regla acorde con dicho tipo de habitación. Esto conlleva múltiples declaraciones sobre conceptos, restricciones y relaciones. La persona que defina un cambio de entorno, debería definir los nuevos tipos de habitación y eso requiere que esta persona tenga conocimientos sobre representación de reglas lógicas y dedique parte de su tiempo en la redefinición de reglas. Esto se ve en robots como el que describen en [157], donde aunque el usuario puede añadir información en la base de conocimiento del robot mediante un tour, las reglas deben estar predefinidas, como en [44].

En este sistema, la semántica del un entorno puede ser cambiada rápidamente al no tener que dedicar tiempo a escribir las reglas lógicas que definen el entorno, por ejemplo pasar de un entorno doméstico de un hogar al de una oficina. Esto hace que el navegador semántico propuesto en esta tesis pueda ser mantenido y utilizado por un usuario sin conocimientos en lenguajes de representación de conocimiento, además de ahorrar el tiempo de definir reglas al usuario que sí posea dichos conocimientos. También será más sencillo que el robot pueda completar la información de las tablas de forma autónoma en función de los objetos que percibe, como se verá cuando se trate al capítulo de adquisición de nuevo conocimiento. Las relaciones entre entidades están implícitas en el diseño de la base de datos, por lo que el cambio de entorno se reduce a introducir datos simples en la base de datos. Para definir el entorno de una oficina, se añadirían los objetos, las utilidades, los tipos de estancia y el resto de conceptos descritos en la sección de modelado del entorno. Si, por ejemplo, un despacho es el lugar que contiene un ordenador y por lo tanto es un buen lugar para trabajar porque los ordenadores se utilizan para ello, es algo que quedará implícito en la descripción del entorno y así lo entenderá el sistema de inferencia, aunque no haya habido ningún humano que haya definido una regla al respecto. Además, esta arquitectura también permite que el robot pueda deducir nuevos tipos de estancia sin crear nuevas reglas. La idea de [43] de mantener por un lado una jerarquía de entidades abstractas o conceptuales y, por otro lado, los objetos físicos percibidos, se aplica también a esta tesis. Ambas jerarquías se mantienen relacionadas. Los beneficios del sistema utilizado

en esta tesis se consiguen tanto con bases de datos convencionales como con bases de datos orientadas a objetos. En este trabajo se han usado bases de datos convencionales porque permite separar la información de los objetos que debe ser almacenada de los datos que no deben ser guardados. Además, las bases de datos orientadas a objetos simplifican las relaciones *ES-UN* pero en este modelo también se utilizan relaciones de otro tipo. Además, la información relativa a unir elementos físicos y conceptuales, y añadir información semántica al objeto funcionan mejor con modelos relacionales.

Una aparente desventaja de este sistema de razonamiento es la rigidez que podría suponer a la hora de definir entornos. Tal vez si en algún momento no se cumple que haya relaciones entre objetos y el lugar donde se ubican, el sistema no sería tan útil. Sin embargo, en todas las civilizaciones del mundo desarrollado se mantiene el mismo patrón a la hora de definir entornos humanos. Desde que el ser humano empezó a construir herramientas, los artefactos tuvieron una utilidad. El sistema propuesto permite establecer relaciones entre objetos y estancias, así como entre objetos y otros objetos. Al no estar estrictamente especificada la naturaleza de estas relaciones, será igualmente válido para cualquier entorno donde los objetos se relacionen con estancias o con otros objetos. Si alguna vez el sistema se somete a un entorno donde no exista ninguna de estas relaciones, la navegación no se vería beneficiada.

Esta arquitectura también permite que el robot pueda deducir nuevos tipos de estancias, sin crear reglas nuevas.

4.1.1. Representación del modelo relacional

El modelado semántico tiene la necesidad de representar el conocimiento en un modelo relacional. En la figura 4.3 se muestra el diagrama entidad-relación que se ha empleado. Los objetos conceptuales y las estancias físicas se relacionan con objetos físicos y estancias físicas, siguiendo las ideas propuestas por otros autores como [157], donde unen la jerarquía conceptual con la jerarquía espacial. En [44] tienen la misma idea y tras extraer los datos sensoriales, realizar la segmentación del mapeo geométrico y extraer la topología, recurren al etiquetado semántico con estructuras que denominan *anchor* como en la figura 4.4. En la Figura 4.5 se puede ver otro ejemplo de las relaciones conceptuales del mismo trabajo, las cuales definen parte de la ontología que emplearon.

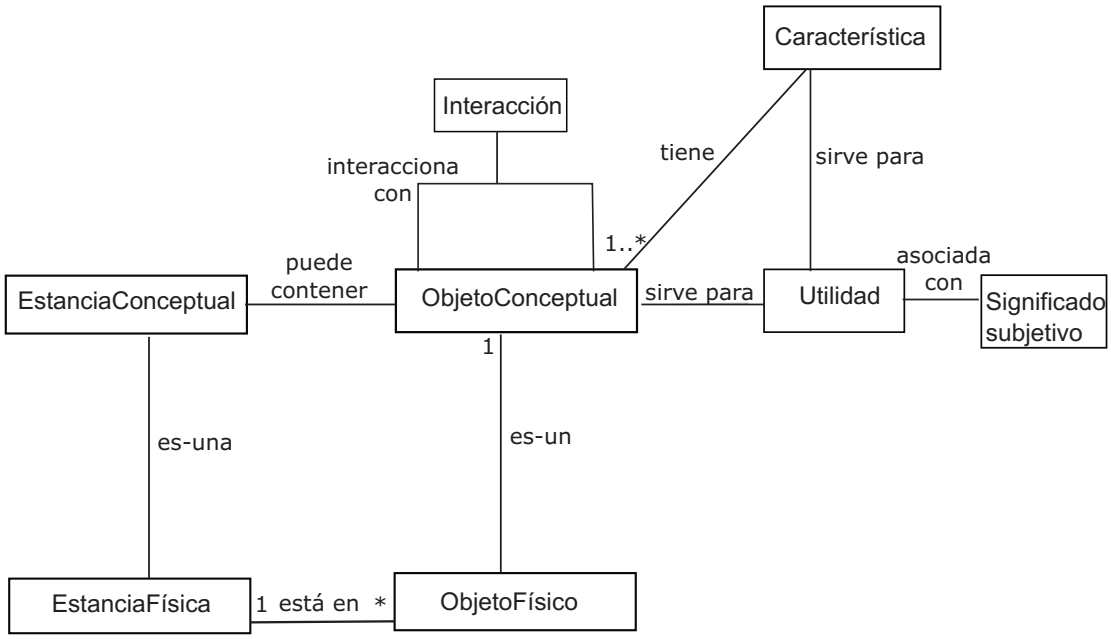


Figura 4.3: Diagrama entidad-relación del modelo ontológico.

Se han considerado los objetos de un entorno doméstico. Esto se refleja en otro modelos de conocimiento como el que usan en [63]. En ese trabajo relacionan objetos con verbos mediante la relación *usarPara* (usedFor), que es similar a la relación *utilidad* que se ha empleado en esta tesis. Este modelo (Figura 4.3, se comenta más adelante) recoge los principales aspectos de la representación semántica del entorno propuesta, cuyos lugares, objetos y significados son almacenados en tablas. El modelo propuesto se centra en la tarea de navegación y desde ese punto de vista se ha diseñado intentando que sea lo más completo y simple posible. Otros modelos de conocimiento semántico más genéricos (no centrados en navegación) son similares al modelo descrito en [74] (Figura 4.6). Como se puede ver, en ese trabajo se requiere un modelo universal que gestione eventos, recursos, actores, contextos, etc. Otros autores como [133] se basan en modelos genéricos como el anterior pero son ajustados para entornos domésticos (Figura 4.7). En ese modelo los conceptos se asocian mediante la relación *es-un* (is-a) y esto genera un modelo aparentemente más complicado que el propuesto en esta tesis puesto que éste se centra únicamente en la navegación. Por ejemplo, en el sistema de esta tesis todo lo que el robot necesita

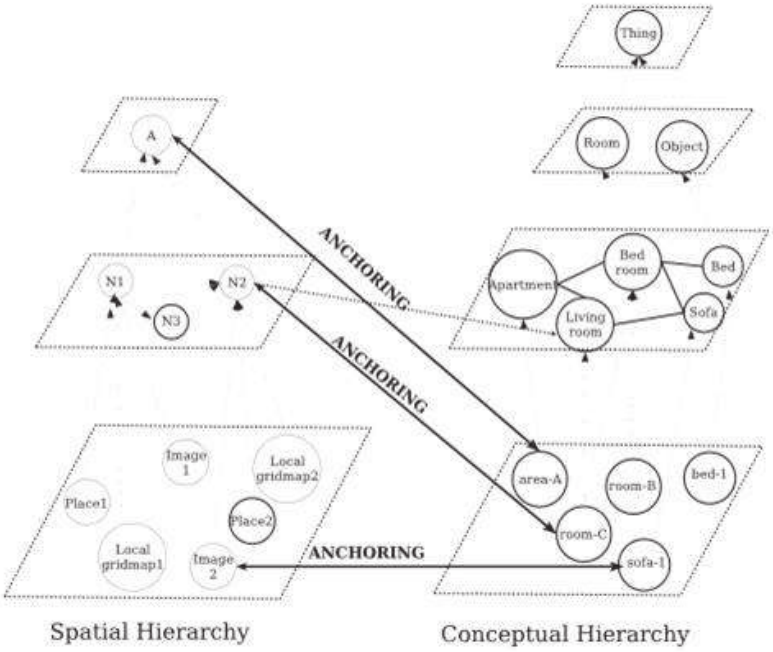


Figura 4.4: Conectividad entre la jerarquía espacial y la jerarquía conceptual. ANCHORING.

saber sobre el lavavajillas es que se encuentra en la cocina y puede contener un cierto número de objetos. Para la navegación es innecesario contemplar que el lavavajillas es un electrodoméstico que hereda de *DispositivoFísico* (PhysicalDevice) el cual hereda de *CosaTangibleSólida* (SolidTangibleThing) la cual hereda de *ParcialmenteTangible* (PartiallyTangible). Por lo demás, el modelo propuesto se utiliza de manera similar al descrito en la Figura 4.7 en el sentido de que se puede almacenar la misma información con la estructura existente. Por ejemplo, si se tienen las entidades *electrodoméstico* y *lavavajillas* como objetos conceptuales, la relación *ES-UN* (is-a) se puede representar con un elemento de la tabla interacción llamado *ES-UN* que una *electrodoméstico* y *lavavajillas*. Por lo tanto el modelo propuesto puede ser utilizado para representar la complejidad deseada (incluso más) pero también para centrarse en la navegación y además simplificar el modelo.

Además, los objetos que se encuentran en entornos domésticos, suelen ser casi en su totalidad objetos creados o modificados por el ser humano con un propósito o funcionalidad. Esto se refleja en otros modelos de conocimiento como se puede ver

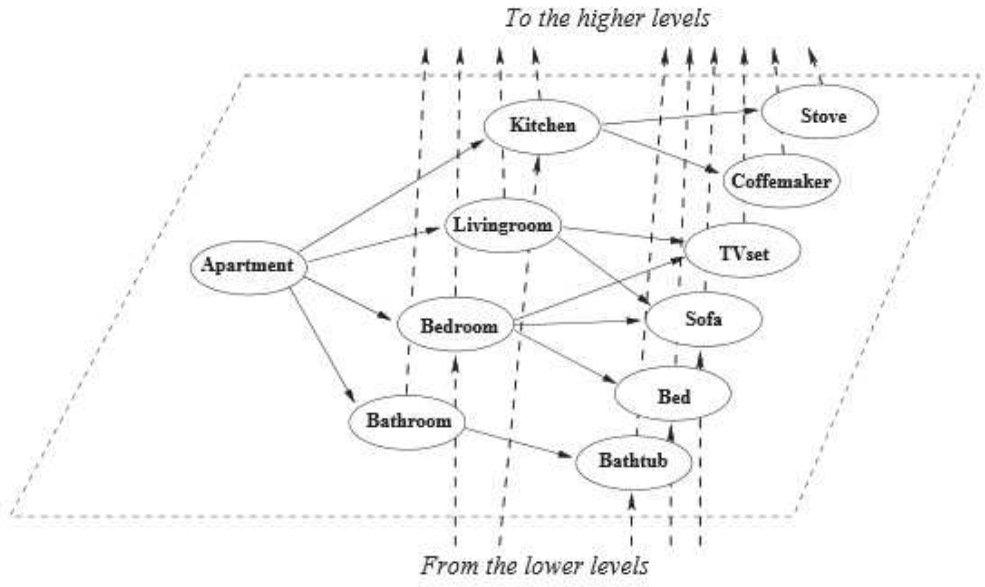


Figura 4.5: Detalle del nivel 1 de la jerarquía conceptual de [44]. Los enlaces horizontales son relaciones tipo *tiene* y las líneas verticales son relaciones tipo *es-un*.

en [63], donde asocian objetos con verbos mediante la relación *usarPara* (usedFor). En esta tesis, se ha denominado el propósito o funcionalidad de los objetos como *utilidad*, y ese término es el empleado en el modelo ontológico para referirnos a dicho propósito.

El modelo de esta tesis (figura 4.3) recoge los principales aspectos de la representación semántica del entorno propuesta. Los cuatro elementos que el robot debe almacenar, como mínimo, son:

- **Estancias conceptuales**. Es la información de las estancias entendidas como conceptos. Son todos los tipos de estancia que se pueden encontrar en un entorno determinado. Por ejemplo, en un domicilio las estancias que se pueden encontrar son el dormitorio, el lavabo, la cocina, el salón, etc.

- **Objetos conceptuales**. Es la información de debe almacenarse sobre objetos entendidos de forma abstracta como conceptos. Son todos los objetos que están definidos en un entorno, entre los que se incluyen los que el robot es capaz de

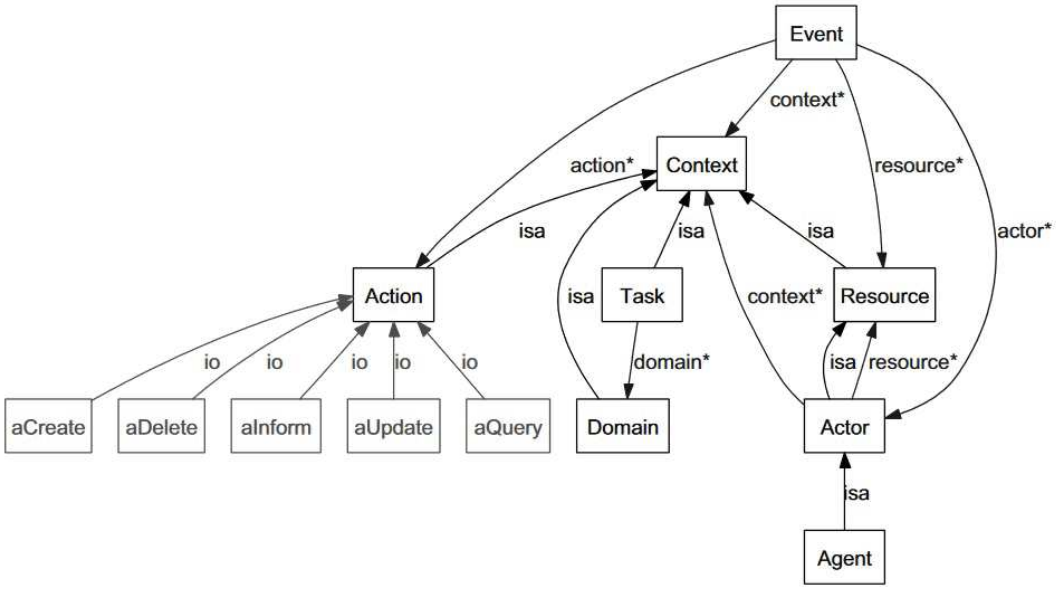


Figura 4.6: Modelo de conocimiento genérico.

detectar. Siguiendo con el ejemplo de un entorno correspondiente a un domicilio, los objetos podrían ser el frigorífico, ordenador, lavadora, sofá, etc.

- **Estancias físicas**. Localizaciones reales sensorialmente percibidas por el robot. Son las habitaciones que reconoce el navegador de bajo nivel, se corresponden con regiones existentes del espacio, zonas delimitadas por paredes, con coordenadas reales.

- **Objetos físicos**. Son los objetos percibidos por los sensores del robot. Principalmente por técnicas de visión que son capaces de reconocer por contornos o por descriptores, un objeto en la imagen que está recibiendo el robot. Cuando un objeto es identificado, se almacena en la base de datos junto con un identificador de posición. La lista de objetos físicos son todos los objetos reales que el robot ha detectado.

Con estas cuatro entidades mencionadas, se han desarrollado varios sistemas de navegación semántica. Sin embargo, como se puede ver en el diagrama del modelo ontológico utilizado, en el navegador propuesto en esta tesis se contemplan, además,

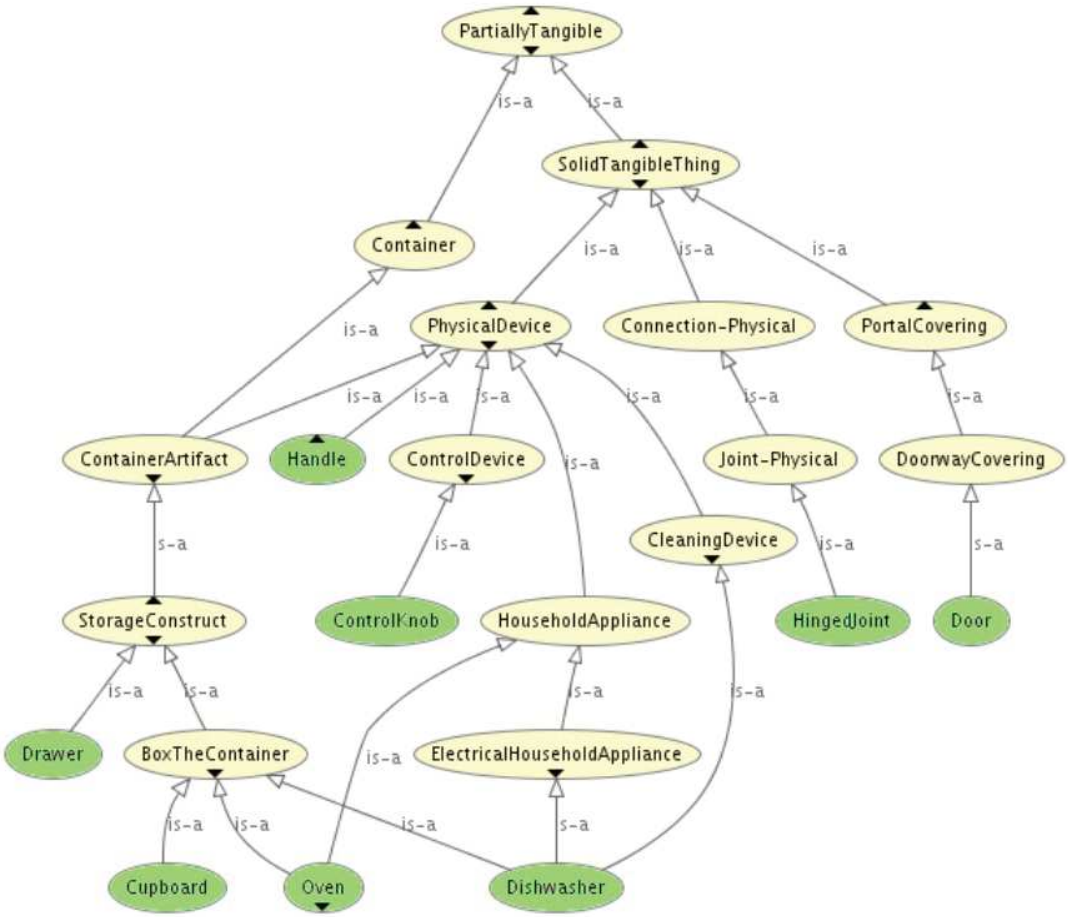


Figura 4.7: Modelo de conocimiento de [133]

unas entidades más que se describen en la sección 4.1.2. Se convierten en una serie de tablas que gestionan información útil que puede ser usada por el robot para encontrar objetos que tienen alguna relación con otros objetos. La idea es que si el robot busca un objeto específico en el entorno del cual no conoce su localización, puede deducirla en función de otros objetos con los que esté relacionado el primer objeto. Se entiende que las relaciones que se tienen en cuenta son las que pueden aventurar una proximidad en las localizaciones. Es decir, suponemos que el objeto A está relacionado con el B en un tipo de relación que puede significar que el objeto A esté próximo al objeto B. Si el robot recibe la orden de buscar el objeto A pero no sabe dónde encontrarlo, buscará primero dónde puede encontrar el objeto B.

4.1.2. Adición de información semántica a los objetos

La figure 4.3, como se ha presentado en secciones anteriores, muestra el diagrama del modelo relacional diseñado. Contiene varias tablas que manejan información concreta que puede ser usada por el robot para encontrar objetos que se relacionan con otros objetos. La idea es que si el robot está buscando un objeto específico del entorno del cual no conoce su localización, la puede deducir por proximidad con otros objetos con los que se relacione. Y entonces puede buscar por otros objetos para encontrar el paradero del primero. Por ejemplo, si el robot busca una impresora es probable que esté cerca de donde se pueda encontrar un ordenador. Así que si el robot no tiene información sobre la ubicación de la impresora, buscará la ubicación del ordenador o de los objetos que se puedan relacionar con la impresora. Por este propósito, se han considerado los siguientes conceptos:

- **Interacción**. Los objetos interaccionan con otros objetos. Este concepto se tiene en cuenta para gestionar cualquier tipo de interacción, aunque en las pruebas pertinentes sólo se haya tenido en cuenta un número limitado de ellas. Las interacciones que han formado parte de las pruebas han sido: SE_USA_CON, cuando un objeto se utiliza con otro objeto y ESTA_DENTRO_DE, cuando un objeto normalmente se puede encontrar en el interior de otro objeto. Esto permite al sistema encontrar objetos que se relacionan de alguna manera porque normalmente se encuentran en localizaciones próximas entre sí. El tipo de interacción es intencionadamente inespecífico, para dotar al sistema de mayor flexibilidad. Decimos que es inespecífico porque se puede incluir cualquier tipo de interacción que el usuario considere que influye en que las localizaciones de dichos objetos también estén cercanas. Esto permite que cualquier usuario de cualquier cultura pueda indicar qué objetos se relacionan de alguna manera, para cualquier relación que provoque que dos objetos estén en localizaciones próximas. Por ejemplo, si de repente surge alguna moda en alguna cultura que diga que todos los objetos del mismo color deben estar guardados en el mismo lugar, el sistema funcionaría si se añade la relación MISMO_COLOR y se indica el dato del color de los objetos.

- **Utilidad**. Se tienen en cuenta los objetos en entornos domésticos que han sido creados o modificados por humanos. Por lo tanto, estos objetos se consideran artefactos y los artefactos por definición han sido creados o modificados para cumplir con un propósito o función. Ese propósito es lo que hemos llamado *utilidad*. Todos los objetos tienen una o más utilidades y la tendencia es agrupar los objetos que tienen la misma utilidad (o alguna relacionada) en los mismos lugares. Además, los tipos de estancia se definen en función de los objetos que contienen y de para qué se utilizan. Por ejemplo, es muy común que todas las herramientas se guarden en el mismo sitio, porque sirven para lo mismo: trabajar. Y un lugar con muchas herramientas tiene muchas posibilidades de ser un taller o un laboratorio de robótica, pero en cualquier caso suele ser un lugar que sirve para trabajar.

- **Significado añadido**. Se ha decidido dotar a las utilidades de un valor añadido subjetivo. Este valor es intencionadamente inespecífico para que el usuario lo gestione a través del diálogo con el robot. De este modo, el usuario añade su percepción subjetiva (una emoción, un sentimiento privado, etc) y lo asocia a una *utilidad*. Se ha llamado a este concepto abstracto de valor subjetivo añadido *significado* para nombrarlo de alguna manera en los diagramas e implementaciones. De este modo, el destino de la navegación también puede ser expresado por una emoción y ser un lugar que produce un sentimiento individual, como ir a un sitio *tranquilo* o ir a un lugar *divertido*. Por ejemplo, si el usuario dice al robot que leer le parece *tranquilo*, el robot asumirá que los lugares donde se puede leer son tranquilos. Así pues, una petición posterior de destino puede ser *ir a un lugar tranquilo* y el robot iría a una sala de lectura.

- **Característica**. Los objetos pueden tener características que definen mejor el concepto o incluso puede diferenciarlos de otros objetos. Por ejemplo, por un lado, el agua puede estar fría, caliente o templada. Y eso implica diferencias en cuanto a su localización. El agua fría se podrá encontrar en un frigorífico o en una fuente, pero el agua caliente podría salir de un fregadero. Estas características han sido tenidas en cuenta en las situaciones en las que los objetos pueden cambiar su funcionalidad cuando cambian sus propiedades, con lo que

```
mysql> select * from estancia_objeto_conceptual;
+------------------+----------------+------------------+
| nombre_estancia | nombre_objeto | numero_aciertos |
+------------------+----------------+------------------+
| Kitchen          | Fridge         |                0 |
| LivingRoom       | Chair          |                0 |
| Office           | Chair          |                0 |
| Office           | Computer       |                0 |
+------------------+----------------+------------------+
4 rows in set (0.02 sec)
```

Figura 4.8: Ejemplo simplificado de contenido en *EstanciaObjetoConceptual*

```
(createConcept Kitchen
   (and Room
        (atLeast 1 stove)
        (atLeast 1 coffee-machine)
        (and (atMost 0 bathtub) (atMost 0 sofa)
             (atMost 0 bed) (atMost 0 tvset))))
```

Figura 4.9: Definición de cocina usando Neoclassic

puede ser que también cambien su localización. Esto es independiente de si la característica es física o abstracta.

4.1.3. Tablas surgidas por la cardinalidad de las relaciones

La información almacenada en el modelo ontológico es la que se emplea para alcanzar los destinos solicitados. Por lo tanto, todas las entidades se corresponden con tablas, pero surgen nuevas tablas al ser implementado el modelo en una base de datos relacional. Es por esto por lo que hay más tablas que entidades (una tabla por entidad, más las tablas que surgen debido a las relaciones muchos-a-muchos del diseño de la base de datos). La base de datos completa (con todas las tablas) que corresponde a esta ontología es la de la figura 4.10.

1. **ConceptualObjectRoom**. Es la tabla que surge de la relación entre EstanciaConceptual y ObjetoConceptual. Es una relación muchos-a-muchos porque un tipo de objeto puede estar en varios tipos de estancia, como por ejemplo una silla que puede estar en salones o despachos. Además, un tipo de estancia puede tener muchos objetos relacionados. Es interesante destacar que parte del conocimiento semántico equivalente a reglas de un motor de razonamiento, está en esta tabla. En la figura 4.8 se muestra un ejemplo de un entorno simplificado. La tabla muestra que la estancia despacho tiene los objetos silla y ordenador, la estancia cocina tiene el objeto frigorífico y la estancia salón tiene el objeto silla. Por lo tanto, si se percibe una silla, el sistema no puede deducir el tipo de estancia en el que se encuentra, pero si percibe un ordenador sí deduce que sólo puede estar en el despacho. Del mismo modo, si percibe un frigorífico deduce que la estancia es una cocina. En contraposición, en sistemas de razonamiento como Neoclassic vistos en trabajos como en [44], se emplean reglas como la de la figura 4.9 para definir una cocina. Ese sistema de reglas exige que el robot haya percibido completamente el entorno antes de poder deducir el tipo de estancia en el que se encuentra, puesto que aunque haya percibido un hornillo tiene que asegurarse de que no hay ningún televisor, ya que la regla no se cumpliría. Además exige también que se predefinan antes todos los tipos de estancia y se especifique lo que deben contener o no. En el método de esta tesis, se introducen los objetos que se pueden encontrar en cada tipo de estancia y será la propia distribución de los objetos los que definen el lugar. Si el frigorífico o un hornillo son objetos que sólo se pueden encontrar en un tipo de estancia concreto, esto se reflejará en la tabla. Y cuando se perciba ese objeto, se podrá concluir que se está en ese tipo de estancia.

2. **PhysicalConceptualRoom**. Esta tabla relaciona las estancias conceptuales con las estancias físicas. Es el efecto del etiquetado. También es una relación muchos-a-muchos porque este sistema contempla la posibilidad de que una misma localización se corresponda con varios tipos de estancia. Como, por ejemplo, un salón que también es comedor.

3. **InteractionCharacteristicObject**. Se tiene en cuenta una relación entre interacciones y características de los objetos. Esto es porque si un objeto está asociado con una característica, puede cambiar su interacción con otro objeto. Por ejemplo, el vinagre es un producto que si se utiliza para el consumo, es probable que esté en un armario de la cocina. Y la relación sería entre el vinagre y el armario de cocina del tipo ESTA_DENTRO_DE, que veíamos en las tablas anteriores. Sin embargo, hay vinagres que se venden como producto de limpieza y podría estar guardado en un cuarto de limpieza.

4. **CharacteristicObject**. Esta es la tabla que relaciona las características con los objetos conceptuales.

5. **UtilityObject**. Los objetos, como se ha repetido varias veces, tienen asociado un propósito o utilidad. Sin embargo, es una relación muchos-a-muchos porque un mismo objeto puede tener varias utilidades, así como una utilidad puede ser cubierta por varios tipos de objeto. En esta relación se añade un atributo extra, que es el *grado*. Esto es porque se puede diferenciar o evaluar de alguna manera subjetiva, lo bien que se adecúa un objeto a una utilidad. Por ejemplo, aunque una silla, un sofá y una cama sirven para descansar, no aportan el mismo descanso. O lo que es lo mismo, no sirven igual de bien para el mismo propósito. Este valor es aportado por el usuario mediante diálogo, y se han establecido tres niveles para no complicar demasiado la interacción y dar un valor discreto y acotado a dicho grado.

6. **MeaningUtility**. La relación entre utilidad y significado añadido también es del tipo muchos-a-muchos, puesto que muchas acciones pueden aportar lo mismo y el mismo valor añadido puede ser aportardo por muchas acciones. Por ejemplo, acciones como *jugar, leer, ver televisión* pueden estar relacionado con *divertido*. Por otro lado, significados añadidos como *didáctico* o *divertido* podrían estar asociados a la misma acción de *leer*.

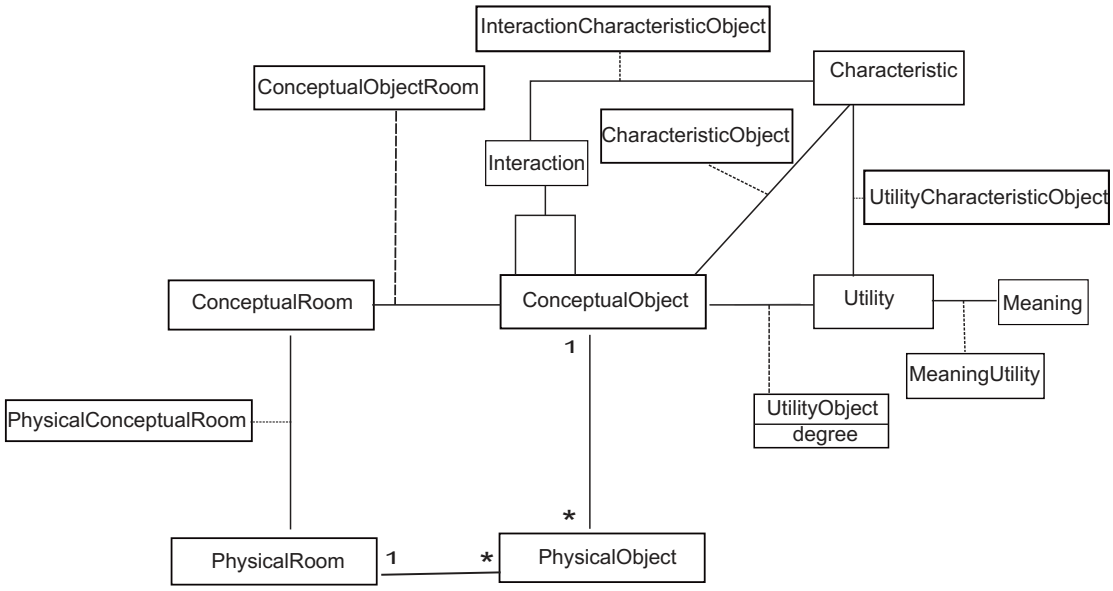


Figura 4.10: Implementación de la ontología propuesta en una base de datos.

4.1.4. Conectividad entre elementos conceptuales y elementos reales

Una vez definida la ontología e implementada en la base de datos relacional, hay que aclarar cómo queda el proceso de conexión entre los conceptos abstractos y los elementos físicos del entorno. El sistema tiene dos tipos de elementos del mapa semántico que se corresponden con elementos tangibles o localizaciones físicas.

El navegador de bajo nivel, tanto el geométrico como el topológico, incluyen objetos en sus mapas. La tabla *PhysicalObjects* contiene cada uno de estos objetos del entorno que han sido detectados. La conectividad con la jerarquía conceptual viene de la relación entre esta tabla con *ConceptualObjects*. Como puede verse en la figura 4.10, la relación es muchos-a-uno, pues se pueden detectar muchos objetos del mismo tipo (muchas sillas, por ejemplo) pero cada objeto detectado únicamente se corresponde con un objeto conceptual. Por ello esta información es almacenada únicamente en la tabla *PhysicalObjects*. Otros autores usan relaciones para unir la información simbólica con los elementos del entorno, como en [28] donde usan correspondencias entre lo simbólico y los datos de los sensores de objetos físicos.

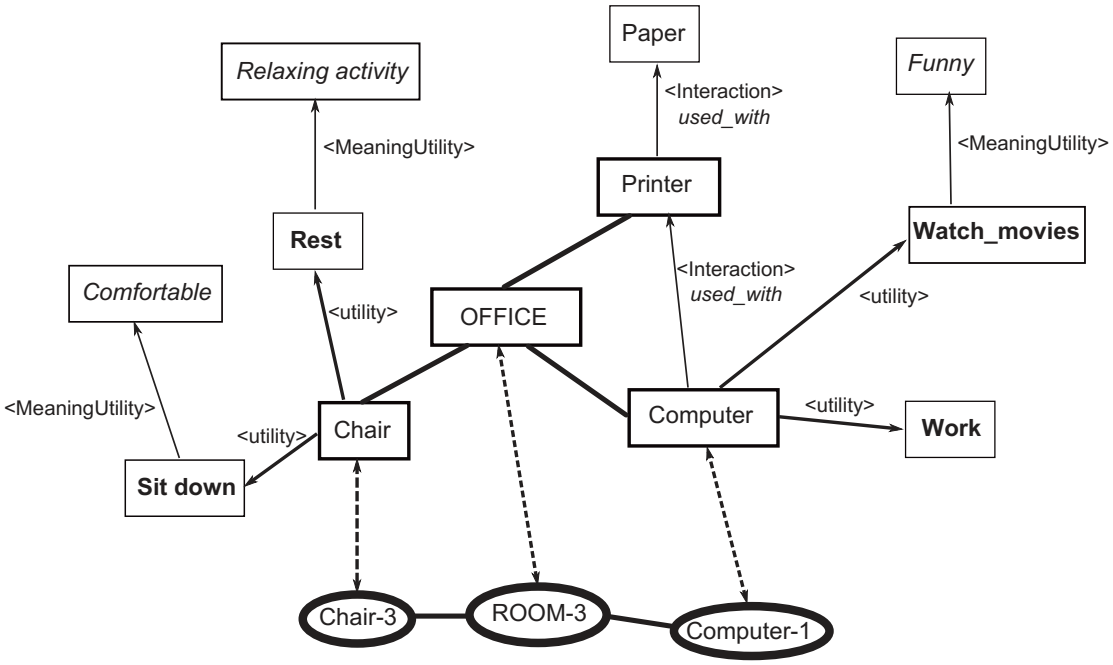


Figura 4.11: Ejemplo de las relaciones entre elementos conceptuales y físicos en un entorno parcialmente explorado.

El otro elemento de la jerarquía espacial, aparte de los objetos, es el relativo a las localizaciones o habitaciones detectadas. Cada habitación detectada tiene una posición en el navegador de bajo nivel. Si se trata de un navegador geométrico, la posición es una coordenada y si es un navegador topológico, la posición es un nodo. En cualquier caso, son localizaciones físicas del entorno real, por ejemplo la *estancia-1* o la *estancia2*. Esta información es la que se almacena en la tabla *PhysicalRoom*, la cual se relaciona con *ConceptualRoom* mediante la relación muchos-a-muchos que se ha descrito en la sección anterior.

Con este modelo relacional implementado en una base de datos, el conocimiento semántico se obtiene lanzando una serie de consultas sobre dicha base, sin la necesidad de definir reglas como han sido descritas en otros trabajos [34][35]. Para visualizar mejor las dos jerarquías que surgen con esta implementación, en las figuras 4.11 y 4.12 se muestra cómo la información podría quedar organizada tras haber explorado parcialmente un entorno. La figura 4.11 muestra las relaciones que el robot conoce sobre la estancia-3. En este ejemplo, la estancia-3 es un despacho. Las entidades de la

jerarquía conceptual están representadas por cajas rectangulares, cada caja se refiere a un elemento conceptual de la ontología. Por otro lado, las entidades de la jerarquía espacial están representadas por elipses, cada elipse se corresponde con un objeto o lugar físico. Las relaciones entre elementos del nivel conceptual se corresponden a la arquitectura ontológica. El conocimiento semántico se manifiesta en la unión de la jerarquía conceptual y la jerarquía espacial. La información que se muestra en esta figura quiere decir que la impresora está relacionada con la silla del despacho y el ordenador. Esto es porque son los objetos que están en el tipo de estancia *despacho*.

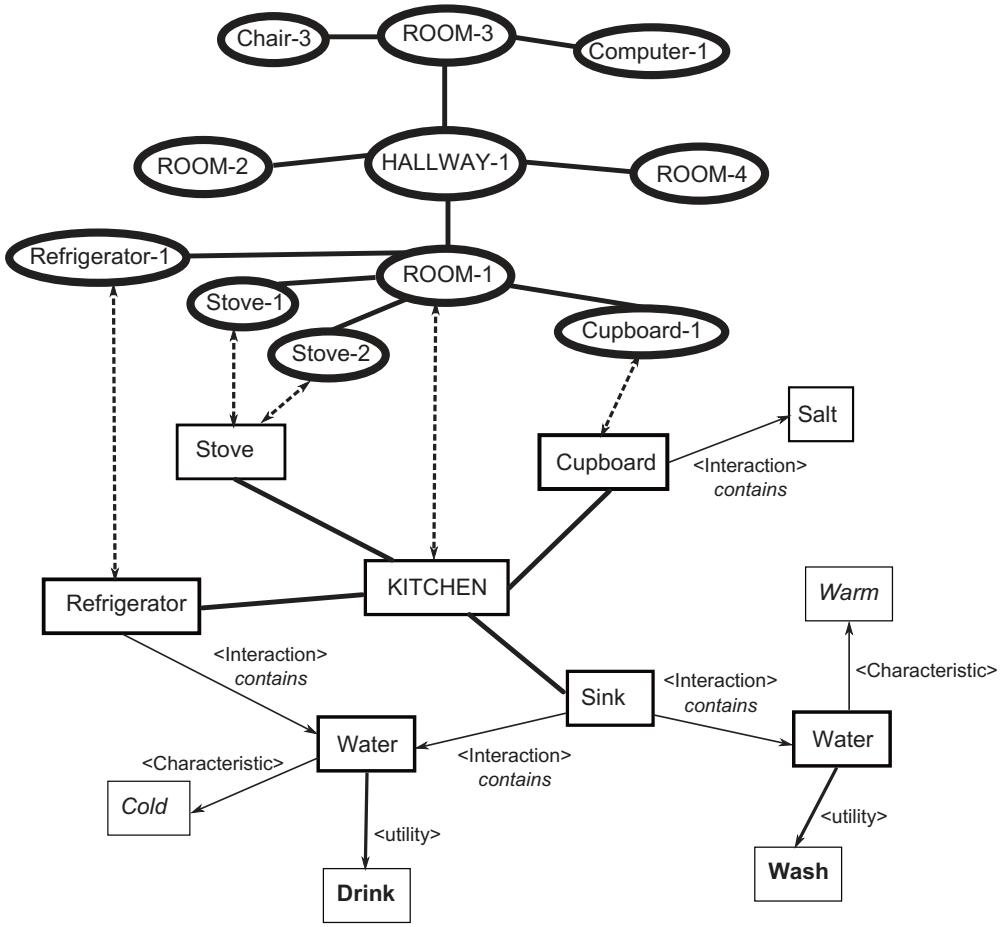


Figura 4.12: Ejemplo de las relaciones entre elementos conceptuales y físicos en un entorno parcialmente explorado II.

A su vez, estos objetos se relacionan con otras entidades u objetos. La impresora se relaciona con el papel mediante la interacción SE_USA_CON. El ordenador se relaciona con *trabajar* y *ver películas*, pues son dos utilidades que tiene dicho objeto y que se contemplan en la ontología. Además, *ver películas* está relacionado con el concepto *divertido*, que se corresponde con el tipo de relación entre utilidades y significado añadido. El ordenador interacciona con la impresora porque las impresoras se usan con ordenadores. Por ello, aparece la relación entre *ordenador* e IMPRESORA mediante la interacción SE_USA_CON. La silla sirve para sentarse y descansar, es por esto por lo que aparece en el diagrama de la figura con estas dos utilidades. Estas dos utilidades se relacionan, respectivamente, con *cómodo* y *actividad relajante*.

En la figura 4.12, lo que se puede ver es una ampliación de la jerarquía espacial, la cual está conectada con otra parte de la jerarquía conceptual. Para no sobrecargar el diagrama, la conectividad de la estancia-3 (el despacho), que correspondía con la figura 4.11, ha sido omitida. Ahora, se muestra la unión de la estancia-1 con el nivel conceptual. Esta estancia ha sido conectada con el concepto *cocina*, puesto que se identifican dos hornillos, un frigorífico y un armario de cocina. Esta figura muestra un tipo de relación que no estaba en la figura anterior y se corresponde con las características de los objetos. Mientras que el frigorífico únicament contiene agua fría, el fregadero puede contener agua fría o caliente. Y de acuerdo a esta característica del agua, su utilidad cambia. Esto se ve en la figura porque el agua con la característica *fría* se relaciona con la utilidad *beber*, mientras que el agua con la característica *caliente*, se relaciona con la utilidad *limpiar*. Como se puede ver, la base de datos contiene información de la ontología del robot, de los objetos físicos detectados en el entorno y de las estancias físicas identificadas en el mapa.

Considerando los elementos de las dos figuras comentadas, las tablas de la base de datos contendrían la información que se puede ver en la figura 4.13. Esta figura muestra el contenido que queda en la base de datos que se corresponde con el entorno descrito del ejemplo.

Partiendo de esta situación, se plantean a continuación dos ejemplos hipotéticos. Suponiendo que un usuario quisiera ir a un lugar divertido, el sistema extrae del modelo ontológico la información de que debe ir al lugar donde se encuentre el objeto

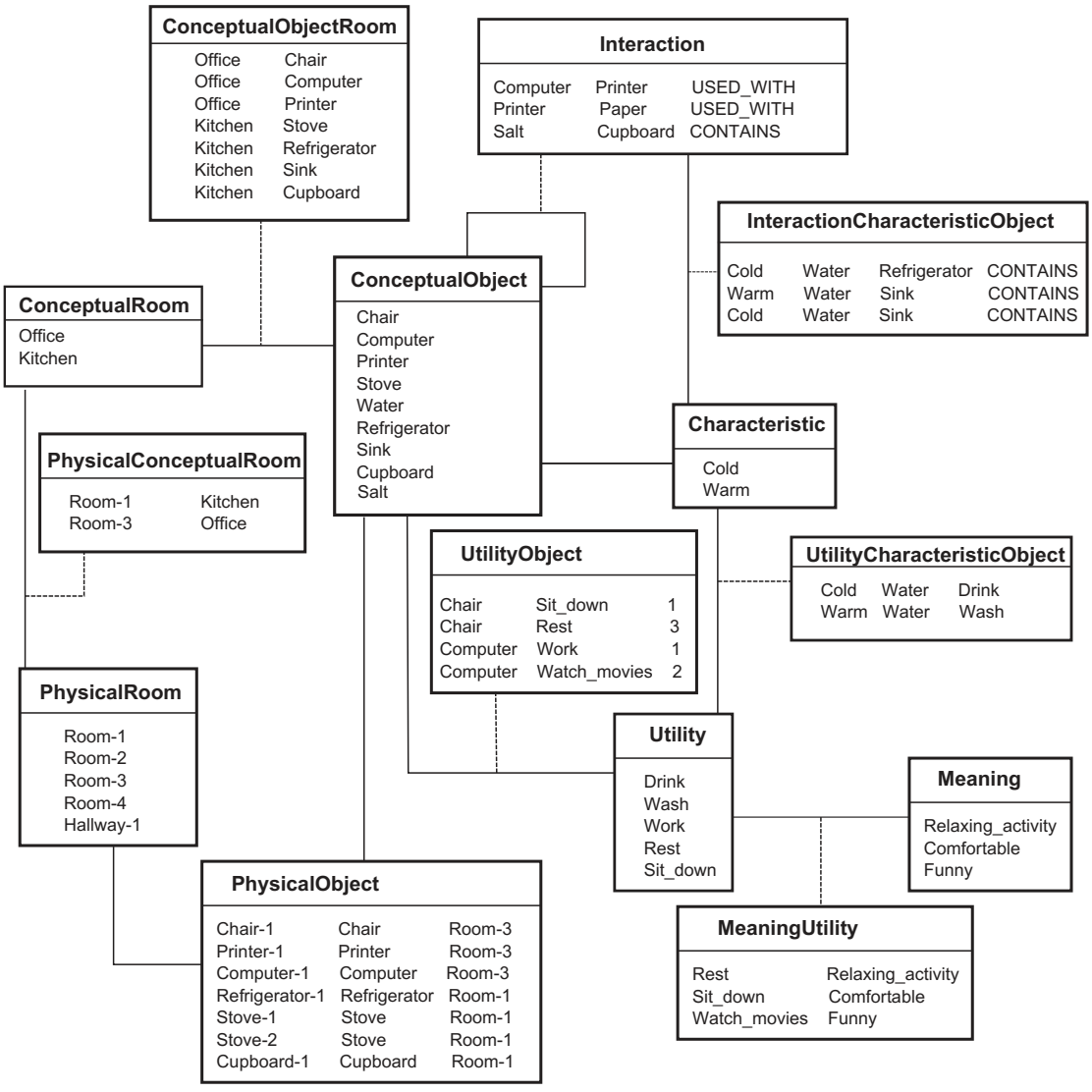


Figura 4.13: Instancia de la base de datos correspondiente al ejemplo del entorno parcialmente explorado.

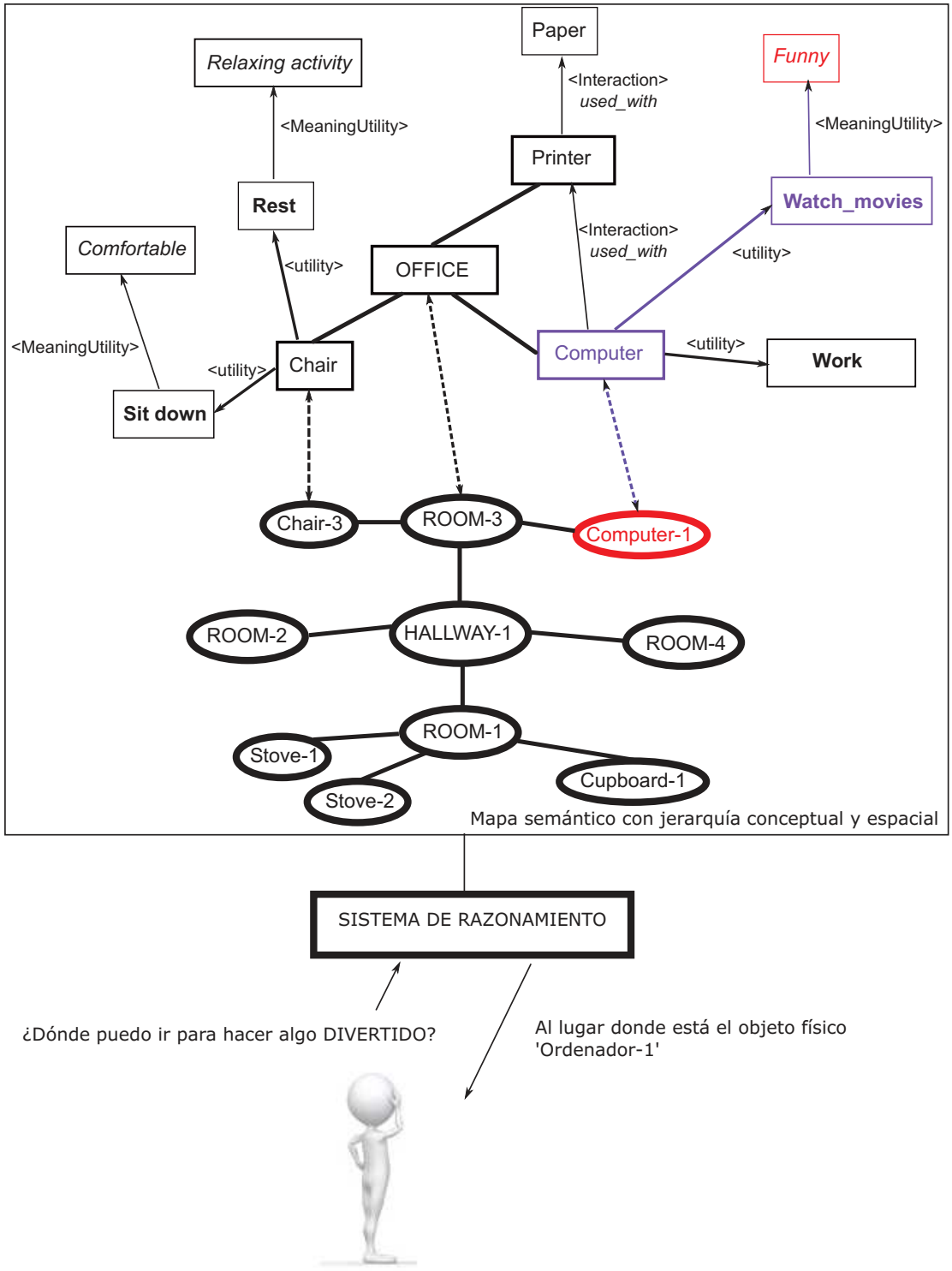


Figura 4.14: Ejemplo de obtención de destino cuando el destino es un objeto.

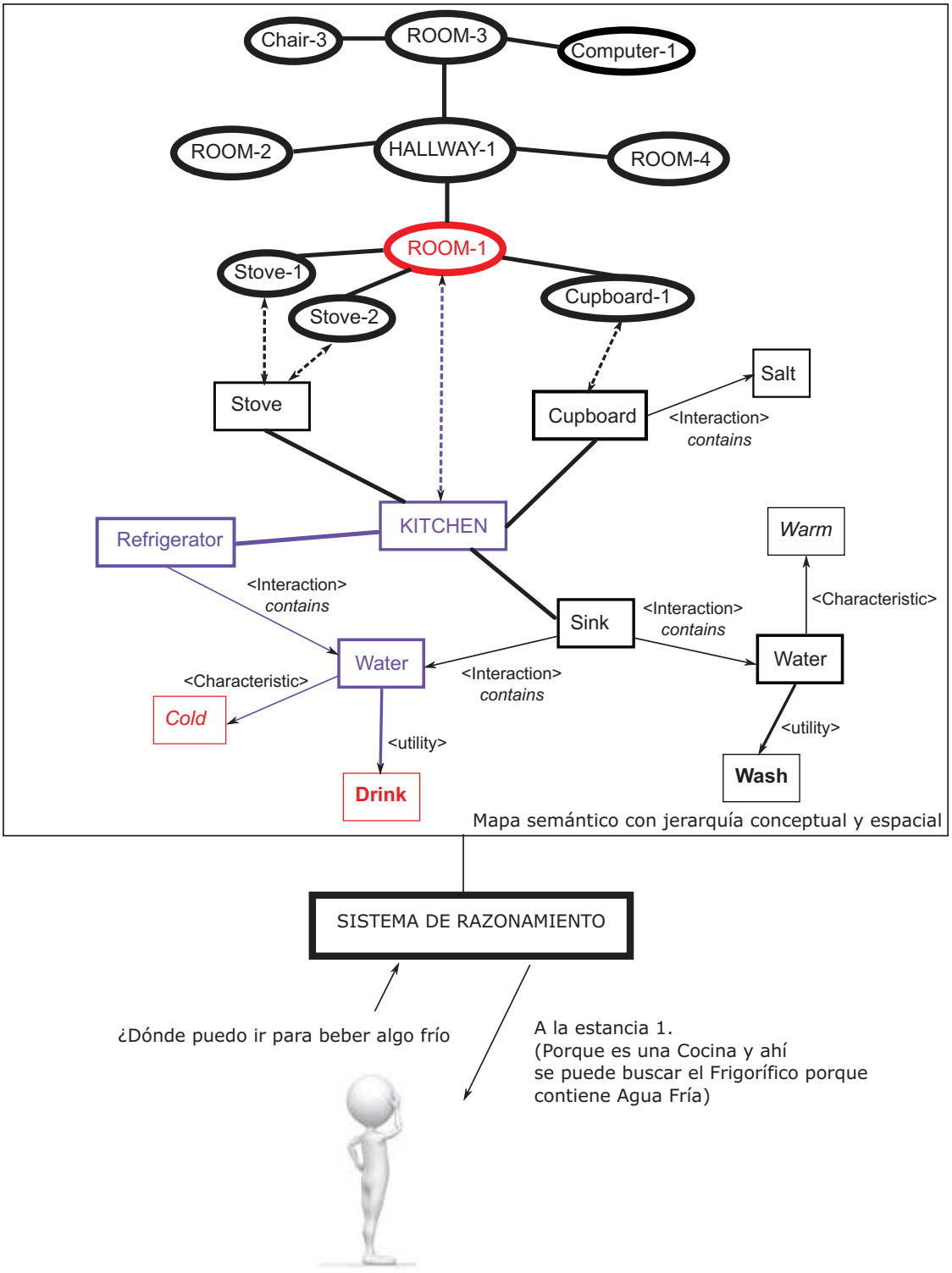


Figura 4.15: Ejemplo de obtención de destino cuando el destino es una estancia.

Ordenador1 como muestra la figura 4.14, puesto que al ser un objeto físico ya percibido con anterioridad, el navegador de bajo nivel conoce su posición y no necesita nada más para llegar hasta ahí. En otro ejemplo, si prescindimos del objeto percibido *frigorífico* y un usuario solicita ir a un lugar donde pueda beber algo frío, el sistema concluye que debe ir a la estancia 1, porque es una cocina y en el modelo ontológico se muestra que en las cocinas puede haber frigoríficos y estos pueden contener agua fría. Esta deducción se trata de ilustrar en la figura 4.15.

4.2. Arquitectura del sistema de inferencia

El sistema de inferencia es el componente clave del módulo de razonamiento y lo que otorga al robot la inteligencia necesaria para cumplir sus objetivos. Es el subsistema que permite al robot extraer conclusiones y manejar la información semántica. Este elemento puede afectar en gran medida al rendimiento y la eficiencia del navegador, por lo cual se ha tratado de facilitar al máximo su integración y su sustitución por otros sistemas de inferencia. Esto tiene como objetivo permitir la comparación de distintos mecanismos de razonamiento y con ello tener el sistema actualizado al mejor o el más adaptado a una necesidad concreta.

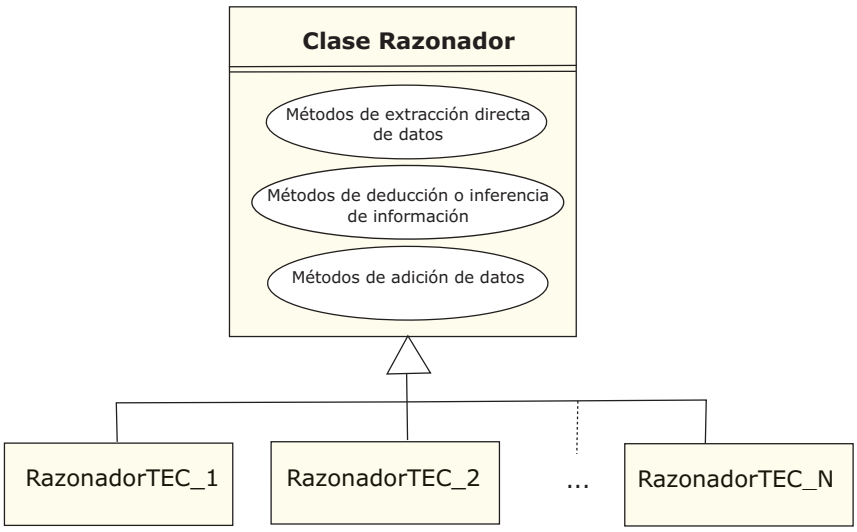


Figura 4.16: Contenido heredado de la clase Razonador.

El sistema de inferencia consta de una clase que hereda de la superclase Razonador. En esta superclase se encuentran declarados todos los métodos que necesitan el resto de sistemas, los cuales son implementados usando la tecnología pertinente en cada clase heredada, como se representa en la figura 4.16. La parte importante de este punto es que independientemente de la tecnología empleada la integración será inmediata si hereda de esta superclase.

4.2.1. Superclase Razonador

Los métodos que posee la clase Razonador se pueden agrupar en función de lo que ofrecen. Por un lado están los métodos que extraen conocimiento almacenado, los más importantes son los siguientes:

- **ListaObjetosDeUnaEstancia**. Método que devuelve la lista de objetos que han sido detectados en una estancia física (jerarquía espacial).

- **ObjetosFisicosXConcepto**. Método que devuelve todos los objetos físicos y percibidos por el robot en el mundo que corresponden a un objeto conceptual. Su salida es un vector *string* con todos los objetos de la base de datos que son reconocidos del tipo de la entrada. Si la entrada era SILLA y no había característica o tenía el valor "NULO", la salida podría ser SILLA-1,SILLA-2 y SILLA-3. Esto significaría que el robot reconoce esos 3 objetos como silla. En la base de datos estarían guardados y asociados a una posición física.

- **TodosObjetosConceptuales**. Método con el que se obtienen todos los objetos del mundo conocido. Sirve para conocer la lista de objetos que reconoce el sistema.

- **GetNombreConceptualDeObjetoFisico**. Este Método devuelve el nombre conceptual con el que está asociado un objeto físico ya detectado.

Por otro lado están los métodos que deducen información o realizan inferencias en función de los datos que posee actualmente el robot (conocimiento almacenado o percepciones sensoriales):

- **IdentificarEstanciaFísica**. Método que dado una lista de objetos percibidos, identifica los tipos de estancia a los que están asociados y los vinculan al objeto estancia. El objeto *estancia* tiene un campo para almacenar los posibles tipos de estancia que puede ser. Por ejemplo, la función puede recibir una lista de objetos como SILLA, MESA. Entonces el método podría introducir en el atributo "tiposPosibles" los tipos de estancia COMEDOR, SALA_REUNIONES, DESPACHO. Si un objeto identifica seguro un tipo de estancia, se almacena en el atributo "códigosIdentificados". Es decir, si hay un hornillo, el tipo COCINA no es sólo posible, sino que esa estancia es una cocina. Aunque tuviera también mesas y sillas, y fuera posiblemente el COMEDOR, también sería cocina (cocina-comedor).

- **ListaTiposDeEstanciaDeUnaEFísica**. Método que dice lo que es una estancia física. Ejemplo: la estancia-3 es un SALÓN, COMEDOR y COCINA. Devuelve la lista de nombres conceptuales de los tipos de estancia. La entrada se refiere al nombre físico (la jerarquía espacial) de la estancia.

- **EstanciasConceptualesConElObjetoX**. Método que devuelve la lista de estancias conceptuales en las que se puede encontrar un objeto. Por ejemplo, una SILLA puede estar en SALÓN,COMEDOR,DESPACHO.

- **ObjsRelacionadosSemanticmnt**. Método que devuelve todos los objetos conceptuales que están relacionados de algún modo con el objeto introducido y su característica. Las asociaciones utilizadas hasta ahora han sido SE_CONTIENE_EN y SE_USA_CON. Sin embargo el nombre de la asociación es irrelevante, este método sólo busca que exista alguna relación. Este método tiene dos entradas: *objetoConceptual*, el nombre del objeto del que se buscan sus relaciones y *caracteristica*, si el objeto tiene una característica concreta, se indica. Si no, este campo queda vacío o con el texto "NULO". La salida es un vector de *strings* con todos los objetos relacionados. Si el objeto relacionado está en un único lugar posible, se indica con un string con este formato: NombreObjeto—NombreLugar.

  Por ejemplo:

  - Se introduce PAPEL y se obtiene IMPRESORA.

  - Se introduce AGUA FRÍA y se obtiene FRIGORÍFICO—COCINA, FREGADERO—COCINA, FUENTE—PASILLO.
  - Se introduce IMPRESORA y se obtiene ORDENADOR.

- **HayObjetosConCaracteristica**. Método que comprueba si existe la asociación de un objeto y una característica.

- **ObtenerObjetosXAccion**. Método para obtener los objetos que sirven para una determinada acción/ tienen determinada utilidad.

- **ObtenerObjetosXAccionYCar**. Método para obtener los objetos que sirven para una determinada acción/ tienen determinada utilidad y que tienen una determinada característica

- **ObtenerObjetosXSignif**. Método para encontrar objetos que cumplen acciones con un significado añadido. Por ejemplo, objetos que hagan cosas que sean DIVERTIDO. El método podría identificar que las acciones JUGAR, LEER y VER_PELÍCULA están relacionadas con el significado añadido DIVERTIDO. Por lo tanto podría devolver la lista de objetos CONSOLA, ORDENADOR, LIBRO, TELEVISION porque sirven para esas acciones.

- **UbicPosibleDeUnObjeto**. Método que devuelve la ubicación posible de un objeto conceptual, con una característica que puede ser nula o no. Por ejemplo, si la entrada fue AGUA como objetoConceptual y FRÍO como característica, los *string* que tendría la salida vendrían en este formato: NombreEstancia—NombreContenedor, siendo el NombreContenedor opcional. La salida podría ser la siguiente lista: COCINA—FRIGORIFICO, COCINA—FREGADERO, PASILLO—FUENTE.

- **EstFísicaXConcepto**. Método que devuelve la lista de estancias físicas que se identifican con una estancia conceptual. La salida es un vector de *string* con todos los lugares de la jerarquía espacial que se corresponden con ese tipo de estancia.

Finalmente, están los métodos que añaden o cambian conocimiento:

- InsertarCaracteristica. Añade una nueva característica para objetos al sistema.

- AsociarCaractObj. Asocia una característica ya existente con un objeto también existente.

- AnadirInteraccionEntreObjetos. Registra una interacción entre dos objetos. Entendiéndose "interacción" como una relación cualquiera entre los objetos que aumenta la posibilidad de que estén cerca el uno del otro espacialmente hablando.

- AnadirUtilidadObjetoCar. Añade una utilidad o acción realizable para un objeto con una característica.

- AsociarSigAccion. Asocia un significado añadido a una acción. Por ejemplo, una asociación del tipo *trabajar* es *estresante*.

- AnadirObjetoFisico. Añade un objeto físico percibido a la base de datos de conocimiento.

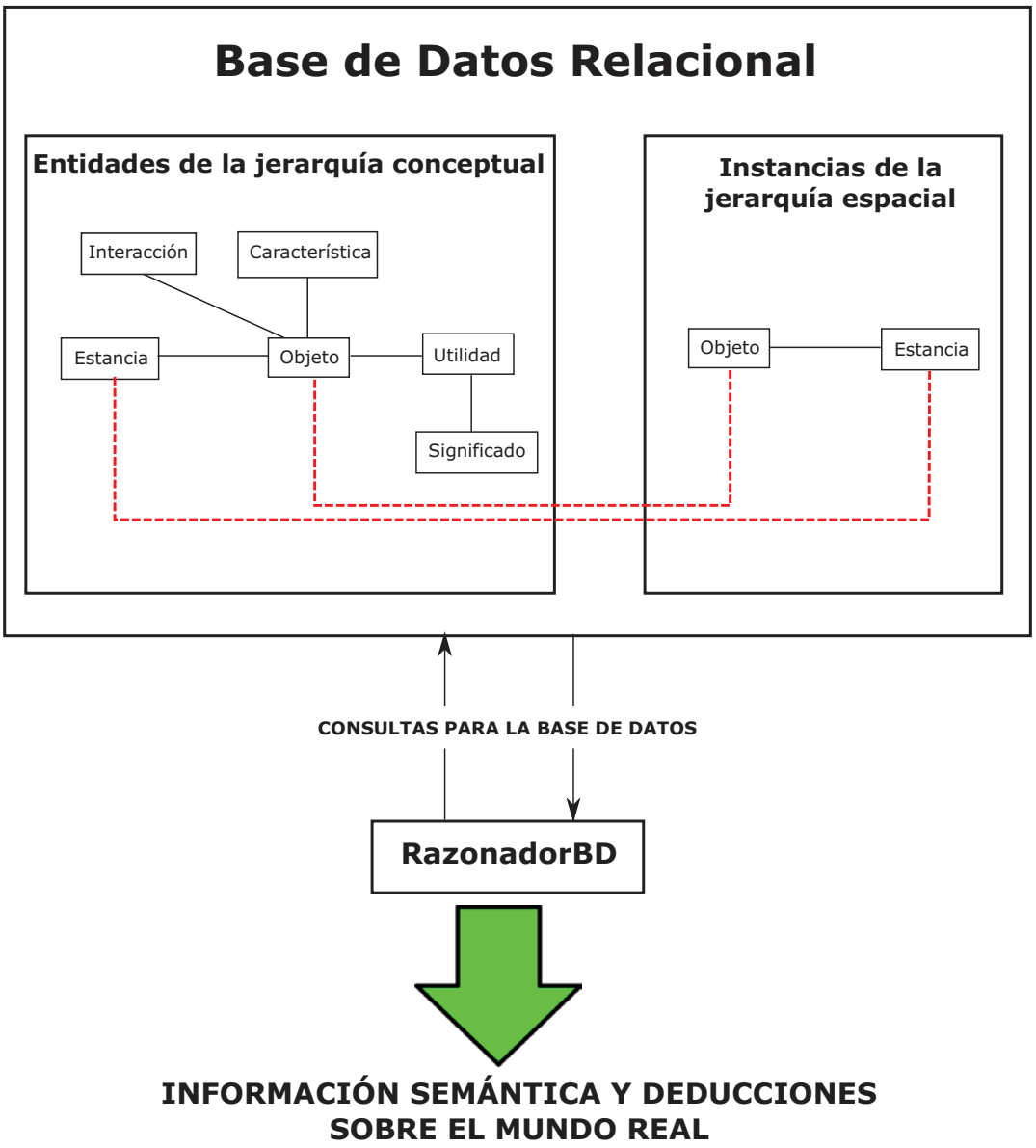


Figura 4.17: El RazonadorBD extrae información mediante una base de datos relacional.

En el desarrollo de esta tesis, se han probado dos mecanismos de razonamiento basados en sendas tecnologías diferentes. El primero de ellos está basado en un middleware creado para esta tesis y basa las implementaciones de la superclase Razonador

en consultas a la base de datos relacional descrita en la sección 4.1.1. El segundo de ellos es un prototipo basado en la tecnología de KnowRob y desarrollado durante una estancia predoctoral en el Instituto de Inteligencia Artificial de la Universidad de Bremen.

4.2.2. Mecanismo de inferencia basado en consultas

El desarrollo del primer mecanismo se inició durante el Trabajo Fin de Master del autor [59], y en los comienzos de esta tesis doctoral fue ampliándose y perfeccionándose, convirtiéndose en el sistema de razonamiento por defecto y sobre el que se han realizado las pruebas de navegación y exploración. Además, su implantación ha producido como resultado las publicaciones [30] y [31]. El concepto básico de funcionamiento parte de la premisa de que si tenemos en una base de datos relacional almacenados tanto las entidades de la jerarquía conceptual como las instancias de la jerarquía física en listas y relacionamos esas listas en un modelo entidad-relación, mediante una serie de consultas SQL predefinidas, se puede extraer la información que necesita un sistema de razonamiento orientado a la navegación semántica. Esta idea se representa gráficamente en la figura 4.17.

Clase RazonadorBD

La clase RazonadorBD hereda de la clase Razonador que se ha visto en el apartado anterior, por lo tanto tiene los mismos métodos descritos anteriormente. Las clases que heredan de Razonador constituyen el middleware que une el sistea de razonamiento empleado con el sistema de navegación semántica. La particularidad de esta clase concreta es que está implementada realizando peticiones de consulta a un servicio ROS que es atendido por otro nodo dedicado a interactuar con una base de datos implementada en MySql.

Por lo tanto, los métodos heredados de Razonador se resuelven invocando como cliente a los servicios de un nodo que efectúa consultas SQL contra la base de datos. Algunos de estos servicios y las consultas realizadas son:

- EstanciaFísicaXconcepto. Es el servicio que lanza una consulta a la base de datos para obtener las estancias físicas (jerarquía espacial) que se corresponden con una estancia conceptual.

```
select nombre_fisico from estancia_conceptual_fisica
where nombre_conceptual ='CONCEPTO_ENTRADA'
```

- ObjetoFísicoXconcepto. Este servicio lanza una consulta para obtener el objeto(s) físico(s) del mundo real que se corresponde(n) con un objeto conceptual que puede tener o no característica.

```
select nombre from objeto_fisico
where nombre_conceptual ='NOMBRE_ENTRADA'
      and nombre_caracteristica='CARACTERÍSTICA_ENTRADA'
```

- EstaciaYContenedorXObjeto. Este servicio consulta en la base de datos para obtener la estancia en la que puede estar un objeto. Además, si el objeto está contenido dentro de otro objeto, también se muestra en el resultado como una respuesta combinada entre el contenedor y la estancia donde está el contenedor.

```
select nombre_estancia, contenedor
from obj_contenidos_y_no2
where nombre_objeto ='NOMBRE_ENTRADA'
```

La tabla obj_contenidos_y_no2 no es una tabla como tal, sino una vista definida de la siguiente manera:

```
create view obj_contenidos_y_no2 as
select nombre_estancia, nombre_objeto,
       null contenedor, numero_aciertos
from estancia_objeto_conceptual
UNION ALL
select estancia_objeto_conceptual.nombre_estancia,
       interaccion_objeto_conceptual.nombre_contenido as
       nombre_objeto,
```

```
        interaccion_objeto_conceptual.nombre_contenedor
        as contenedor, numero_aciertos
from interaccion_objeto_conceptual,
     estancia_objeto_conceptual
where nombre_contenedor=nombre_objeto and
      nombre_interaccion='SE_CONTIENE_EN'
UNION ALL
select nombre_estancia,
        objeto_caracteristica_interaccion.nombre_objeto,
        nombre_contenedor, 0 numero_aciertos
from objeto_caracteristica_interaccion,
     estancia_objeto_conceptual
where nombre_contenedor=
            estancia_objeto_conceptual.nombre_objeto
      and nombre_interaccion = 'SE_CONTIENE_EN'
group by nombre_contenedor,nombre_objeto;
```

- ObjetoXAccion. Este servicio genera una consulta para obtener objetos conceptuales que tienen una utilidad concreta que se introduce como entrada. La lista de objetos que se van a devolver como salida están ordenados por grado, que es una valoración que da el usuario sobre lo bien que se ajusta el objeto a su utilidad. Por ejemplo, para la utilidad *descansar* los objetos podrían ser *silla* con grado 1, *sofá* con grado 2 y *cama* con grado 3.

```
select nombre_objeto, nombre_caracteristica
from obj_car_util_completa2
where nombre_utilidad='UTILIDAD_ENTRADA' order by grado
```

La tabla obj_car_util_completa2 vuelve a ser una vista definida previamente.

```
create view obj_car_util_completa as
select * from objeto_caracteristica_utilidad UNION ALL
select nombre_objeto, null nombre_caracteristica,
nombre_utilidad, grado from objeto_utilidad;
```

```
create view obj_car_util_completa2 as
select * from obj_util_car UNION ALL
select nombre_objeto, null nombre_caracteristica,
nombre_utilidad, grado from objeto_utilidad;
```

- ObjetoXProximidadSemanticaCar. Este servicio hace una consulta que busca en toda la base de datos los objetos que pueden estar relacionados con otros objetos, mediante el concepto interacción, teniendo en cuenta además si tienen alguna característica que modifique esa relación. Las relaciones pueden ser de cualquier tipo, las que se han usado en las pruebas eran de objetos que contienen otros objetos y de objetos que se usan con otros objetos.

```
select nombre_estancia, nombre_contenedor
from obj_cont_y_no_car
where nombre_objeto ='OBJETO_ENTRADA'
  and nombre_caracteristica = 'CARACTERÍSTICA_ENTRADA'
```

Esta consulta emplea varias vistas

```
create view parcial_1 as
select nombre_estancia, nombre_contenido,
       nombre_objeto, numero_aciertos
from interaccion_objeto_conceptual,
     estancia_objeto_conceptual
where nombre_contenedor =
          estancia_objeto_conceptual.nombre_objeto
      and nombre_interaccion='SE_CONTIENE_EN';

create view obj_cont_y_no_car as
select estancia_objeto_conceptual.nombre_estancia,
       estancia_objeto_conceptual.nombre_objeto,
       nombre_caracteristica, null nombre_contenedor,
       numero_aciertos
```

```
from estancia_objeto_conceptual left join
     objeto_caracteristica on
     estancia_objeto_conceptual.nombre_objeto
              = objeto_caracteristica.nombre_objeto
UNION ALL
select nombre_estancia,
       objeto_caracteristica_interaccion.nombre_objeto,
       objeto_caracteristica_interaccion.nombre_caract,
       nombre_contenedor, 0 numero_aciertos
from objeto_caracteristica_interaccion,
     estancia_objeto_conceptual
where estancia_objeto_conceptual.nombre_objeto
  = objeto_caracteristica_interaccion.nombre_contenedor
    and nombre_interaccion='SE_CONTIENE_EN'
UNION ALL
select nombre_estancia,nombre_contenido as
       nombre_objeto, nombre_caracteristica,
       parcial_1.nombre_objeto as
       nombre_contenedor,numero_aciertos
from   parcial_1 left join objeto_caracteristica on
  nombre_contenido=objeto_caracteristica.nombre_objeto;
```

La clase RazonadorBD utiliza más servicios atendidos por el nodo que gestiona la base de datos, pero los más relevantes son los que se acaban de comentar y son suficientes para entender el funcionamiento de este sistema de razonamiento. Las deducciones se hacen extrayendo de la base de datos la información necesaria con las consultas predefinidas.

4.2.3. Inferencia con KnowRob

La inferencia con una ontología basada en KnowRob se ha implementado siguiendo la misma estructura ya definida en el sistema de inferencia del navegador semántico. Knowrob e un sistema de razonamiento autónomo que además permite representar

conocimiento. Incluye técnicas de adquisición de conocimiento y aprendizaje de un sistema físico. Knowrob funciona como un entorno de trabajo (framework) semántico común para integración de información de diferentes fuentes. La ontología es definida con el software de Protégé, descrito en [91], el cual genera un archivo .owl que es cargado posteriormente por un servidor ROS que atiende a las consultas del razonador.

Ontología con KnowRob

El primer paso es trasladar la ontología al nuevo software. La figura 4.18 muestra varios conceptos de la jerarquía conceptual. El árbol que se forma viene determinado por las relaciones del tipo es-un. El concepto primigenio es "Cosa" y de él heredan los demás. El primer nivel lo forman las entidades Característica, Significado, Objeto, Estancia y Utilidad. Como característica se ha introducido Frío, por significado se introduce Divertido, se tiene una serie de objetos, tres tipos de estancia y cuatro utilidades.

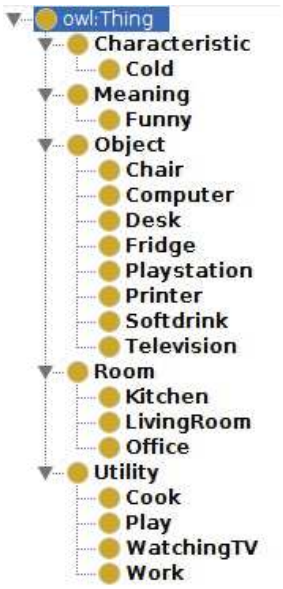


Figura 4.18: Jerarquía conceptual de la ontología implementada en Protege para funcionar con KnowRob.

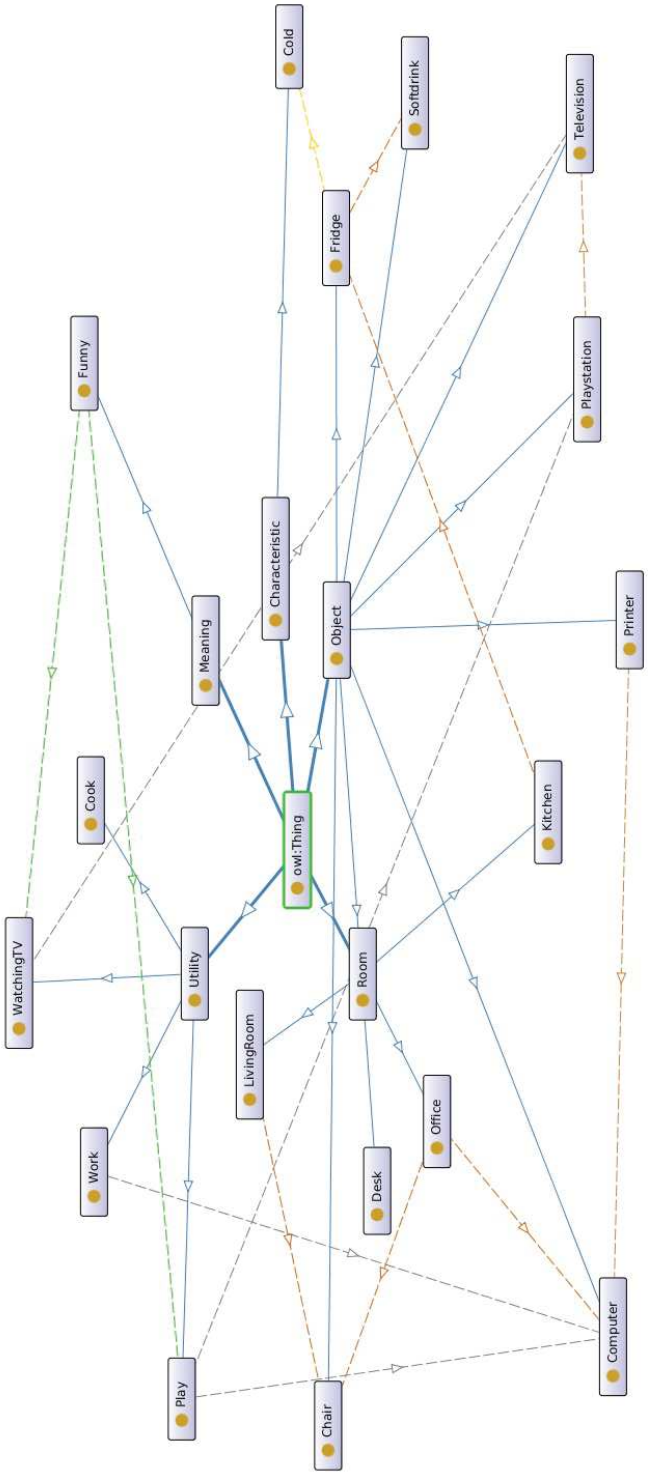


Figura 4.19: Relaciones de la jerarquía conceptual implementada en Protégé.

Sin embargo, para representar conceptualmente las relaciones entre esas entidades, es necesario introducir restricciones y otros tipos de relaciones que se modelan como restricciones. De esta manera, se ha asociado la característica Frío con Frigorífico y todos los objetos contenidos se consideran "Fríos". Para ello es necesario establecer la relación "contener", la cual también se utiliza para especificar qué objetos se contienen en los distintos tipos de estancia. También es necesario relacionar los objetos con su utilidad y las utilidades con significados subjetivos. Así se relaciona, por ejemplo, la televisión con la utilidad "ver televisión" y dicha utilidad con el significado subjetivo "divertido".

Estas relaciones se muestran en el gráfico 4.19, el cual muestra la ontología completa que se ha utilizado en los ejemplos para realizar las primeras pruebas sobre este razonador.

Las relaciones definidas en la ontología son:

- Contains. La relación que indica que una entidad contiene a otra.
- ProducedBy. Esto identifica el tipo de relación donde una entidad es producida por otra.
- UsedWith. Este tipo de relación indica que una entidad es usada con otra.
- ActionPerformedWith. Esta relación es entre acciones y objetos, indica que una acción es llevada a cabo mediante un objeto.

Además, se ha creado otro archivo .owl aparte para la jerarquía espacial, es decir, para los objetos y estancias reales. En la figura 4.20 se puede ver el gráfico que asocia entidades con instancias. En las pruebas se han considerado dos estancias que se corresponden con una cocina y un despacho. La estancia 1 (el despacho) contiene dos sillas y un ordenador. Mientras que la estancia 2 no contiene nada.

Clase RazonadorKR

La clase RazonadorKR hereda de Razonador, del mismo modo que lo hace la clase RazonadorBD. Por lo tanto, tiene los mismos métodos que están descritos en la sección 4.2.2. La diferencia en esta ocasión es que se implementan accediendo a la

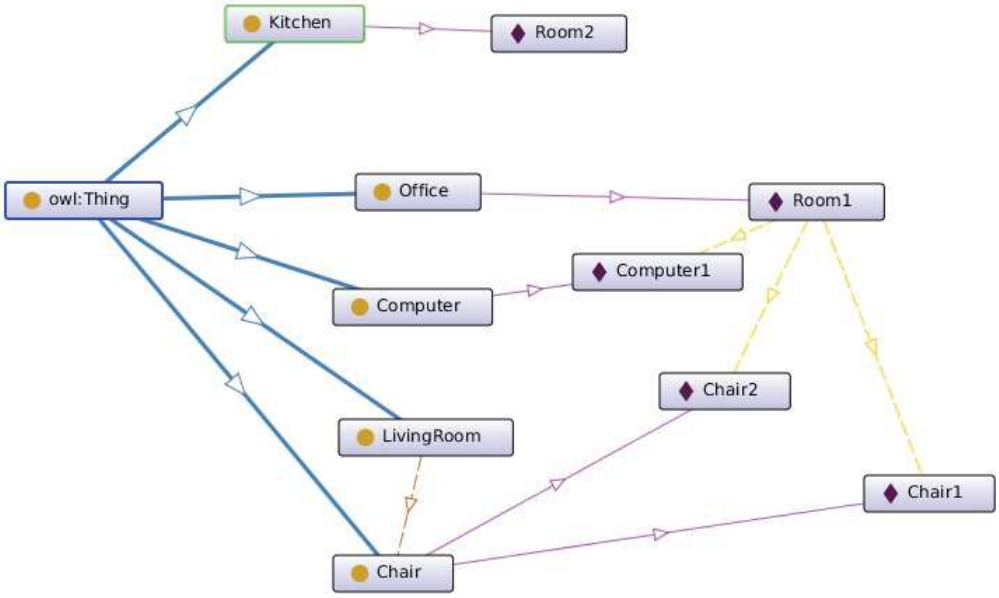


Figura 4.20: Relaciones de la jerarquía conceptual y física implementada en Protege.

ontología almacenada en un archivo .owl con mediante un servidor que funciona con prolog.

Se han empleado tres arquitecturas de consultas diferentes, solicitando la ejecución en el servidor de distintas instrucciones de prolog.

- Consultas de subclase, relaciones is-a. Es la consulta que devuelve la lista de entidades que son subclases de otra entidad, por lo tanto cumplen la relación is-a. Se usa en los métodos todosObjetosConceptuales, todasCaracterísticas, todasUtilidades.

  Por ejemplo, el método todosObjetosConceptuales solicita al servidor todas las entidades que son objetos conceptuales haciendo que se ejecute esta instrucción:

  ```
  owl_subclass_of(A, semantic_nav:'Object').
  ```

- Consulta de propiedades. Para consultar sobre otros tipos de relaciones entre entidades que no sean *is-a*, se utiliza la instrucción *class_properties*. De este modo se accede a las representaciones de cualquier tipo de relación conceptual.

Los métodos que emplean esta instrucción son *obtenerObjetosXSignif*, *objsRelacionadosSemanticmnt* y *obtenerObjetosXAccion*.

Por ejemplo, para obtener las acciones que producen un significado subjetivo concreto, se utiliza la siguiente instrucción:

```
class_properties(semantic_nav:'Significado',
                 semantic_nav:'ProducedBy', A).
```

La cual trata de obtener los valores posibles con los que unifica la variable *A*, la cual obtiene todas las entidades que tengan la relación *ProducidoPor* con una entidad que es un significado. Por ejemplo, el significado *divertido* es producido por las acciones *jugar* y *ver la televisión*.

- Consultas de subclase con propiedades. Esta consulta combina la búsqueda de un entidad que es subclase de otra entidad conocida y que además tiene otro tipo de relación con otra entidad establecida como una propiedad que da nombre a la relación. Es utilizada por los métodos identificarEstanciaFísica, estanciasConceptualesConElObjetoX, objsRelacionadosSemanticmnt y ubicPosibleDeUnObjeto.

  Por ejemplo, el método identificarEstanciaFísica solicita al servidor que ejecute esta instrucción:

```
owl_subclass_of(A, semantic_nav:'Room'),
class_properties(A, semantic_nav:'Contains',
                    semantic_nav:'NombreObjeto').
```

  Debido a que su objetivo es ver si un objeto identifica unívocamente a un tipo de estancia, hace una consulta para saber qué tipos de estancia contienen el objeto que se está percibiendo. De ello se encarga la instrucción *owl_subclass_of*, para unificar en la variable *A* todas las entidades que son subclase de *Estancia* y satisfacen por lo tanto la consulta. La coma es equivalente al operador *AND*, con lo que la instrucción completa hace que, aparte de obtener en la variable *A* todas las entidades que son estancia, la variable *A* también satisfaga la relación *Contiene* con la entidad correspondiente al objeto que se está percibiendo (NombreObjeto). Por eso se combina con la instrucción *class_properties*. El resultado

es que si, por ejemplo, el objeto percibido es un hornillo, la lista de valores para *A* sea únicamente la cocina, con lo que ese objeto habrá servido para identificar la estancia. Mientras que si el objeto es una silla, la lista de valores para *A* podría ser despacho, salón, comedor. Por lo que la silla no identifica la estancia.

- Consulta de instancias de conceptos. Se usa en los métodos *listaTiposDeEstanciaDeUnaEFisica* y *listaObjetosDeUnaEstancia*. Se emplea la instrucción *owl_has*. Por ejemplo, para obtener el tipo de estancia conceptual al que está asociado una estancia física se hace que el servidor ejecute la siguiente instrucción:

```
owl_has(semantic_nav_data:'NombreFísico',
        rdf:type,A).
```

  Esto hace que se consulte en los datos de instancias aquellas que se corresponden con el tipo que viene dado en la variable *NombreFísico*. Si el nombre físico es, por ejemplo, *Estancia2* y la estancia 2 está asociada con una cocina, el resultado será *cocina*.

- Consulta para obtener instancias con una propiedad. Esta consulta necesaria para el método *listaObjetosDeUnaEstancia*, devuelve la lista de instancias que cumplen un tipo de relación con otra entidad.

  Por ejemplo, la instrucción

```
owl_has(semantic_nav_data:'NombreEstancia',
        rdf:type, B),
class_properties(B,semantic_nav:'Contains', A).
```

  Esta instrucción permite obtener todos los objetos que están asociados mediante la relación Contiene con una entidad conceptual de estancia de la cual existe una instancia. Por ejemplo, si la estancia 2 es una cocina y la variable *NombreEstancia* es *Estancia2*, la variable *B* será unificada con *Cocina*. La otra parte de la instrucción hace que se obtengan las relaciones *Contiene* de la variable *B* (Cocina), con lo que se obtienen todos los objetos conceptuales que pueden estar en la cocina. De este modo, el sistema sabría qué tipo de objetos puede haber en la estancia 2.

4.2.4. Ventajas e inconvenientes del razonamiento con bases de datos

El empleo de este sistema de inferencia ha suscitado largos debates sobre su efectividad y características. Llegar a crear estas vistas y consultas que mueven el sistema de inferencia no es un proceso sencillo ni demasiado intuitivo pero es un esfuerzo que se realiza una sola vez y vale para cualquier entorno que se describa con la misma ontología representada en la base de datos relacional. Una vez las vistas están creadas, sirven para cualquier entorno que se introduzca en las tablas de la base de datos.

En el caso de que un usuario quiera utilizar el navegador propuesto, con el sistema de inferencia basado en bases de datos relacionales, simplemente tendría que introducir en las tablas la información de los objetos y sus características, tipos de estancia, tipos de estancia, acciones que realizan los objetos y las interacciones. No sería necesario añadir demasiados datos puesto que el sistema también permite el aprendizaje en tiempo de ejecución. Tampoco se requeriría crear vistas ni tener ningún conocimiento especial en SQL (salvo el de introducir datos, o utilizar algún entorno gráfico para ello).

Esto representa una ventaja esencial sobre el otro sistema de inferencia implementado, la inferencia con KnowRob. Con KnowRob es necesario definir primeramente una ontología en un archivo owl para la que se requiere conocimiento de la herramienta Protege o similar. Además, la instalación de Knowrob no es tan sencilla ni la ejecución tan ligera.

Se planteó el problema de la escalabilidad. Una crítica importante al sistema de razonamiento con bases de datos es la posibilidad de ampliar la ontología, puesto que el diseño de la base de datos es aparentemente más rígido y estático. También preocupó el tratamiento de la recursividad. Ante esto, hay algunos comentarios que realizar. El primero es que no se ha encontrado ningún entorno humano que no cumpla lo siguiente:

- Cada lugar tiene alguna función que define ese lugar. Los lugares de un entorno artificial humano son definidos en función de lo que se puede hacer en ellos.
- Los objetos que cumplen funciones similares, se almacenan normalmente cerca unos de otros.

- Los lugares con una función determinada, tienen objetos que se usan para acciones relacionadas con esa función.
- Algunas acciones son asociadas por los usuarios con algún tipo de emoción, sentimiento o significado añadido.

Por lo tanto, el sistema vale para todos los entornos estudiados hasta el momento, sin necesidad de cambiar la ontología. Por otro lado, en el caso de querer añadir o cambiar algo, el problema también existiría en una ontología implementada con otra tecnología. El esfuerzo de añadir una lista a la base de datos y modificar las vistas y consultas afectadas que permiten obtener la información no es superior al de modificar un archivo owl cambiando o añadiendo relaciones. En este sentido, no se percibe la base de datos como algo estático, puesto que se puede modificar.

En cuanto al asunto de la recursividad, el sistema no plantea ningún problema con ello y se realizaron experimentos cuyo resultado se muestra en la sección 6.1.8. Además, los tiempos de ejecución en el sistema razonador que usa directamente la base de datos son **considerablemente menores** que los tiempos de ejecución del razonador que emplea la tecnología de KnowRob. Los resultados experimentales se muestran en la sección 6.1.9. Los experimentos de comparación entre el sistema de razonamiento que utiliza bases de datos y el sistema de razonamiento que utiliza Knowrob han sido realizados partiendo del mismo modelo ontológico, el cual se ha representado de forma equivalente en los respectivos modelados, por lo que contienen las mismas relaciones entre las entidades conceptuales.

4.3. Obtención del destino semántico

Antes de hablar sobre la obtención del destino semántico, es conveniente aclarar qué es para este sistema un destino semántico. Se ha considerado que cualquier deseo sobre un lugar que tenga el usuario que pueda hacer que ese lugar se diferencie de otro, es un destino semántico, definido por lo tanto como una unidad de información contextual asociada a la posición de un robot. La manera de describir ese lugar es fundamental para el sistema. En este caso, los lugares pueden estar descritos por el tipo de estancia, por el lugar donde se ubica un tipo de objeto u objeto concreto,

por las acciones que se pueden realizar en esa estancia o por el significado añadido que transmita esa estancia. Estos conceptos se guardan en una variable de la clase PosiciónSemántica, y puede usarse para representar la posición actual o la posición destino.

4.3.1. Objetos y estancias como destinos

La clase PosiciónSemántica contiene atributos que corresponden al código de la estancia (la estancia objetivo cuando la variable representa el destino semántico), al código del objeto y al resto de identificadores de circunstancias semánticas del lugar. Se entiende que la posición es obtenida en relación a un punto semántico de referencia, que puede ser un objeto o una estancia. Una vez que el destino al que se quiere dirigir el robot está representado en un objeto de esta clase, el sistema cuenta con varias opciones dependiendo de la información disponible en la base de datos sobre el destino solicitado y el tipo de destino (objeto o estancia).

Estancias desconocidas Si el destino es una estancia desconocida, el campo del identificador de estancia de la variable PosiciónSemántica mantiene un valor nulo y el planificador semántico lanza la correspondiente rutina de exploración semántica para localizar el tipo de estancia solicitado. Aquí se entiende como *estancia desconocida* aquella estancia que el sistema conceptualmente sabe lo que quiere encontrar (por ejemplo, la cocina) pero no tiene constancia de que ninguna estancia física del mundo real del entorno conocido haya sido etiquetada de esa manera. El planificador acaba delegando la tarea en el explorador.

Estancia conocida Si el objetivo es una estancia conocida, significa que la estancia ya ha sido explorada previamente y por lo tanto hay una estancia física a la que dirigirse. La variable destino (un objeto de la clase PosiciónSemántica) en este caso tiene el valor del identificador de la estancia correspondiente con la estancia física ya conocida.

Objeto conocido en estancia conocida Otra posibilidad es que el destino original sea un objeto, por ejemplo el frigorífico. En ese caso, después de completar la función, la variable DestinoSemántico habrá adquirido el identificador del

objeto específico (por ejemplo, frigorífico-1, un objeto del mundo real ya identificado) y la estancia donde se encuentre (por ejemplo, estancia-3). Obsérvese que si el destino es un objeto conocido es porque ya ha sido explorado y significa que la ubicación del objeto también es conocida (el robot tuvo que estar ahí para detectar el objeto) por lo tanto cuando el objeto es conocido, la estancia también es conocida y se puede cargar en la variable DestinoSemántico tanto el objeto como la estancia.

Objeto desconocido en estancia conocida Este es el caso cuando el robot no tiene la información previa de haber percibido un objeto, pero sí ha identificado una estancia física que se corresponde con el tipo de estancia donde debería estar el objeto. En este caso el valor del identificador del objeto de la variable es nulo y el robot explora en la estancia conocida. Por ejemplo, en la estancia-3 si esta es una cocina y el objeto a buscar es el frigorífico.

Objeto desconocido en una estancia desconocida El sistema se encuentra en este caso si el objeto no ha sido percibido previamente y el tipo de estancia donde debería encontrarse tampoco. El procedimiento es primero buscar el tipo de estancia donde es más probable encontrar el objeto y si se encuentra, explorar en esa estancia. Tanto si no se encuentra el tipo de estancia como si se encuentra pero no se detecta el objeto deseado en ella, el planificador decide explorar erráticamente por todo el entorno buscando el objeto.

Para ayudar a entender el proceso de tratamiento de destino, se muestra un ejemplo simplificado en la figura 4.22. Nótese que en este proceso no se contemplan acciones ni significados añadidos, sólo se muestran objetos y estancias dentro de la variable de posición semántica. Se trata de un caso de búsqueda del objeto *silla*. El planificador contiene un vector de destinos que son gestionados en orden LIFO (Last Input First Output) debido a que los destinos son encadenados uno tras otro en el sistema. Esto sucede cuando el sistema se da cuenta de que para alcanzar el primer destino que ha recibido, el robot debe previamente alcanzar otros destinos que van surgiendo. Estos nuevos destinos son añadidos a los ya existentes teniendo prioridad sobre los anteriores. En la figura 4.21 se muestra cómo el sistema va introduciendo destinos parciales

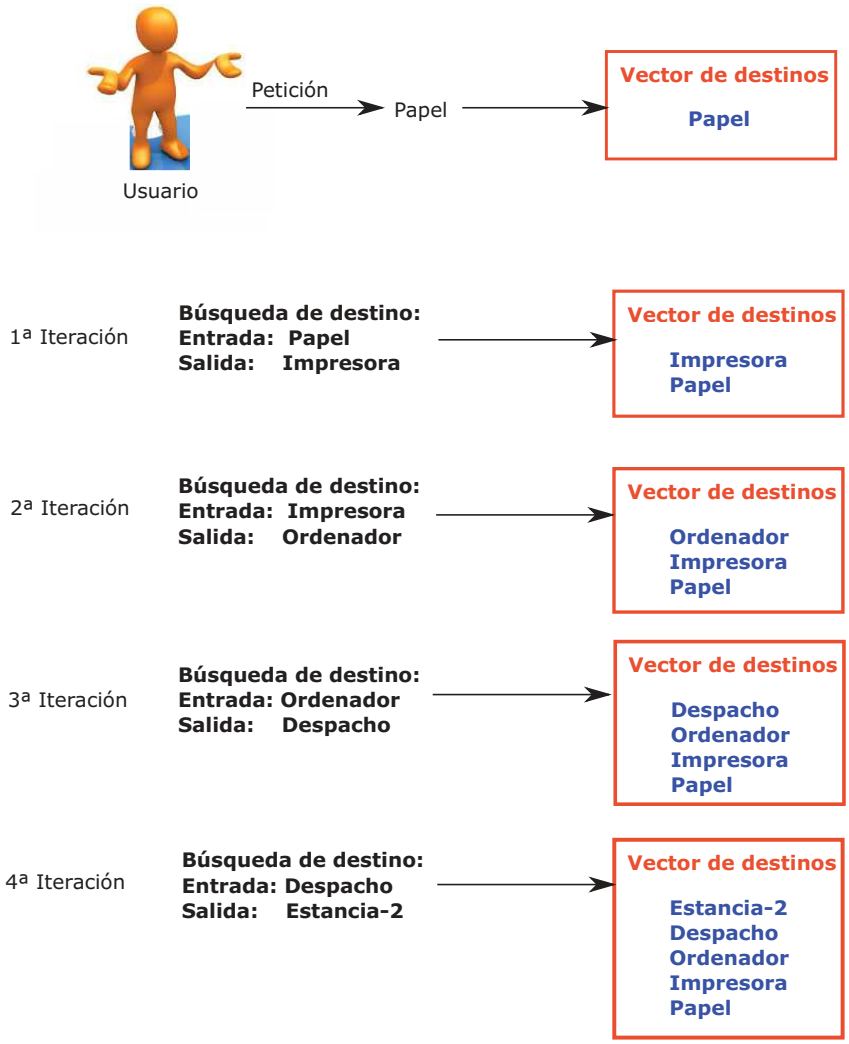


Figura 4.21: Ejemplo de crecimiento del vector de destinos ante una petición por parte del usuario para localizar papel.

según va deduciendo información ante un ejemplo de petición de papel. Para alcanzar el destino principal, debe ir encontrando antes los destinos parciales que se han ido añadiendo en el proceso de localización del destino, el cual se detiene cuando el sistema encuentra un destino interpretable por el navegador de bajo nivel (en este caso, la estancia-2 que corresponde con el despacho).

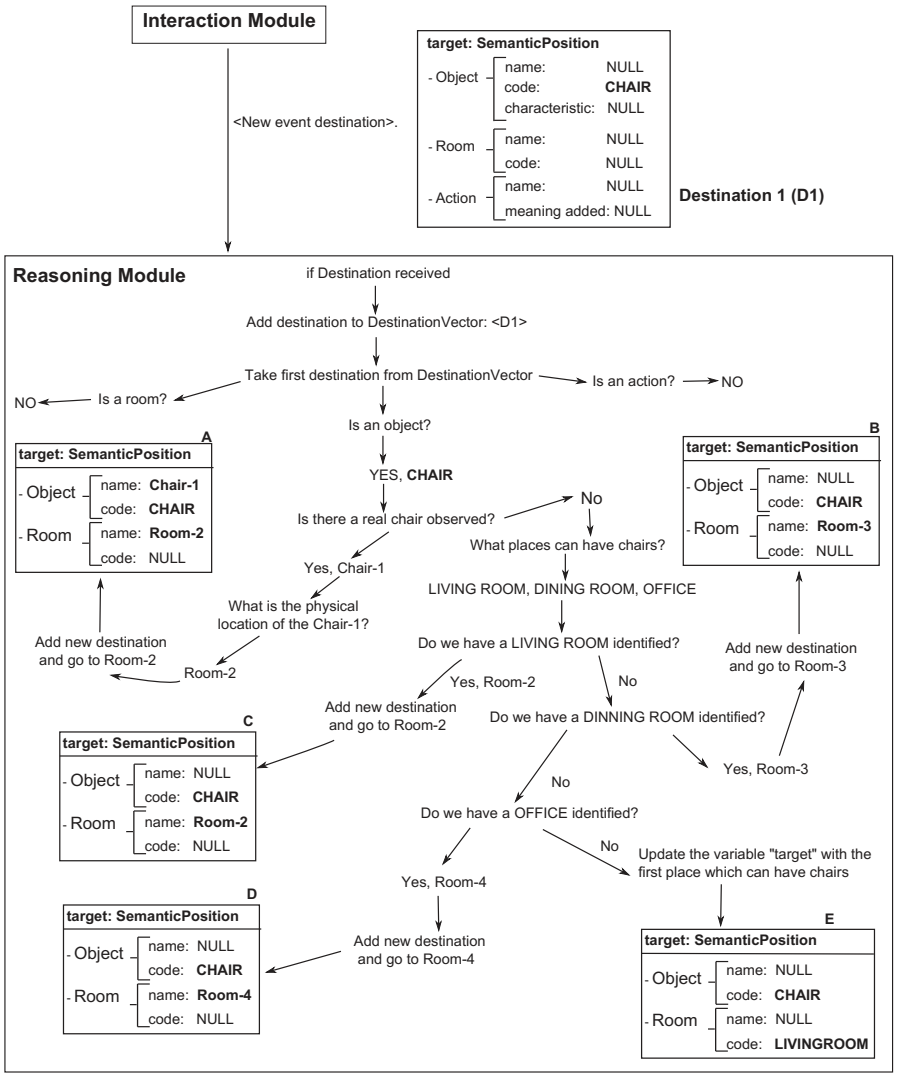


Figura 4.22: Diagrama que muestra un ejemplo conceptual de la función TargetLocation buscando una silla.

Volviendo al ejemplo de localización de un objeto (una silla en la figura 4.22), hay cinco posibles salidas A, B, C, D y E que pueden obtenerse. En el resultado A, la silla es identificada y además la estancia donde el objeto estaba ubicado también ha sido identificada. Cuando el planificador encuentra una situación de este tipo, directamente envía el evento al navegador de bajo nivel para ir a dicha estancia y poder localizar allí el objeto. Sin embargo, para los resultados B, C y D, la variable *Destino* no tiene información sobre ningún objeto físico, sino que la información que posee es la de buscar por el objeto conceptual SILLA y esto además añade información de la estancia física donde el robot debería encontrar la silla. Lo que sucede es que el algoritmo primero intenta localizar una de las estancias donde podría haber sillas. Si la encuentra, hace que el robot se dirija allí y si no la encuentra, hace que el robot empiece a buscar por esos tipos de estancia.

4.3.2. Algoritmo de búsqueda de destino

La parte del sistema de razonamiento que maneja los eventos de peticiones de destino, está implementada empezando por el algoritmo 2. Este algoritmo se ejecuta cada vez que se recibe una petición de destino y hace que el vector de destinos se actualice. En este primer nivel del algoritmo, el objetivo es identificar lo que solicitó el usuario. Las posibilidades son:

- Buscar una estancia física a partir de un tipo de estancia conceptual.
- Buscar un objeto.
- Buscar por una acción (ejemplo: leer).
- Buscar por un significado añadido (ejemplo: algo divertido).

En el caso mostrado en la figura 4.22, puesto que se busca por un objeto, la función que es llamada es *searchForAnObject*, cuyo pseudocódigo es el algoritmo 3. Una observación a tener en cuenta es que tal como se ve en el capítulo 3, la parte que maneja la información semántica está desacoplada del resto del navegador (como todos los módulos). Esto se refleja en la línea 1 del algoritmo, donde se muestra la variable global genérica llamada Razonador. Esta variable es la que asegura que se implementa

Algorithm 2 Target obtention from semantic destination

Require: SemanticPosition *target* with the new target information.

1: **if** *target* is a conceptual room goal **then**
2: getPhysicalRoomFromConceptualRoom(*target*)
3: **else if** *target* is an object goal **then**
4: searchForAnObject(*target*)
5: **else if** *target* is an Action **then**
6: searchForAction(*target*)
7: **else if** *target* is a subjetive meaning **then**
8: searchForMeaning(*target*)
9: **else**
10: print error message
11: **end if**
12: **return**

una interfaz puesto que es iniciada con un objeto que hereda de la superclase Razonador. Esto permite reemplazar el método de inferencia con una asignación a una nueva clase que también herede de Razonador y tenga redefinidos todos los métodos con otra herramienta o tecnología. En cualquier caso, esta parte del sistema extrae la información de los objetos físicos (jerarquía espacial) que corresponden al objeto conceptual (jerarquía conceptual) que está siendo buscado.

Algorithm 3 searchForAnObject destination

Require: SemanticPosition *target* with the new target information.

1: $objectList = reasoner.getPhysicalObjectFromConcept(codigoObjeto, CaracterObjeto)$
2: **if** $listaObjetos \neq \{\}$ **and** existe un objeto-no-descartado **then**
3: publicar un evento con la orden de ir al primer elemento de la lista.
4: **return**
5: **else**
6: searchPosibleLocationFromAnObject(*target*)
7: **end if**
8: **return**

Siguiendo la secuencia de ejecución del ejemplo, si el caso fuese que el sistema encuentra sillas físicas percibidas, la función saldría por la línea 3 y se mandaría la ubicación del objeto físico correspondiente como destino al navegador de bajo nivel. Por lo tanto este escenario es de tipo A en la figura 4.22. Si esta búsqueda falla, se invoca al algoritmo 4. Este algoritmo está diseñado para encontrar todas las relaciones

que tenga un objeto con otros objetos o con estancias con el objetivo de ubicarlo espacialmente.

Algorithm 4 searchPosibleLocationFromAnObject

Require: SemanticPosition *target* with the new target information.

```
1: charact = target.object.characteristic
2: code = target.objetc.code
3: locationList = reasoner.getPosibleLocationFromObject(code, charact)
4: searchSemanticProximity = false
5: if locationList ≠ {} then
6:    bool success = searchObjectsOrRooms(target.object.code, locationList)
7:    if !success then
8:       searchSemanticProximity = true
9:    end if
10: else
11:    searchSemanticProximity = true
12: end if
13: if searchSemanticProximity then
14:    bool result = searchBySemanticProximity(target)
15:    if result = OK then
16:       add new target in destination vector (target updated)
17:    else if target has characteristic then
18:       remove target.object.characteristic
19:       create new target and push on vector
20:       searchPosibleLocationFromAnObject(newtarget)
21:    else
22:       publish event to explore
23:    end if
24: end if
25: return
```

Primero se buscan relaciones directas, de las que relacionan un objeto directamente con una estancia o con otro objeto conocidos por estar contenido en él. Si esta búsqueda tiene éxito, lo siguiente es buscar ese objeto o estancia. Para ello se invoca a la función *searchObjectsOrRooms* que, como se verá a continuación, puede provocar más llamadas recursivas que vayan encadenando destinos. Si esta búsqueda fracasa, se activa la búsqueda por proximidad semántica, en la que se admite cualquier tipo de relación entre objetos y sus acciones que pueda dar pistas sobre el paradero del

objeto deseado. Si esta búsqueda tiene éxito, se actualiza el vector de destinos y esto provoca que se vuelva a llamar recursivamente a la función original con el nuevo destino incluido. Si por el contrario la búsqueda falla, se realiza un intento más pero eliminando la característica del objeto en el caso de que este tuviera. Por ejemplo, si el usuario había pedido un *refresco frío* y no hay ninguno en la nevera, el sistema elimina la característica *frío* y repite la búsqueda semántica. La línea 17 comprueba este resultado e inicia de forma recursiva la búsqueda genérica por el objeto sin característica. Si esto falla, la búsqueda fracasa y se le notificará al usuario.

En el ejemplo de la silla, el flujo del procedimiento lleva a ejecutar la función *searchObjectsOrRooms*, puesto que la lista de localizaciones no está vacía y la línea 6 del algoritmo 4 obtiene una llamada con éxito. El pseudocódigo de esta función es el algorimo 5. Esta función recoge todos los objetos o estancias que van a formar parte de los destinos parciales para alcanzar el objetivo principal. El tratamiento es diferente dependiendo de lo que se haya recibido. Para las estancias, el algoritmo identifica la estancia física que corresponde a la estancia conceptual y crea un nuevo destino parcial con la tupla objeto-estancia. Cuando lo que se recibe es un objeto, el algoritmo llama de nuevo a la función *searchForAnObject*.

4.4. Anticipación al destino

Esta sección está para comentar que en el desarrollo de esta tesis se ha considerado la posibilidad de intentar dotar al robot de la capacidad de anticiparse al destino que pueda necesitar un usuario. De este modo el robot inicia el desplazamiento antes incluso de recibir la orden directa por parte del usuario, tratando de deducir en base a un conocimiento previo el destino que puede ser solicitado. Este enfoque necesita que el robot tenga la capacidad de clasificar situaciones y asociarlas a un destino. La figura 4.23 describe este planteamiento.

La idea es que podría ser posible anticiparse al destino al que quiere ir un usuario si se conocen las circunstancias que identifican que el usuario decida ir a un lugar u otro. Por ejemplo, es posible que el usuario quiera ir al dormitorio cuando quiere dormir, siendo una acción que requiere mucha tranquilidad, a la que quiere dedicar mucho tiempo (8 horas), con una urgencia media, usando el objeto cama, en un

Algorithm 5 searchObjectsOrRooms

Require: String *code*, vector *list*
Ensure: **bool**, **true** if success

```
1: bool OK = true
2: bool continue = true
3: for all elements of list and continue = true do
4:    string object_code = element.object.code
5:    string room_code = element.room.code
6:    if object_code failed previously then
7:       OK = false
8:    end if
9:    if OK then
10:      continue = false
11:   end if
12: end for
13: if OK then
14:   if There is OBJECT then
15:      if There is ROOM then
16:         vectorobjects = reasoner.getPhysicalObjectFromConceptualObject(object_code)
17:         vectorrooms = reasoner.getPhysicalRoomsFromConceptualRoom(room_code)
18:         create newtarget and add it on destination vector
19:      else
20:         searchForAnObject(new SemanticPosition(object_code))
21:      end if
22:   else
23:      reasoner.getPhysicalRoomsFromConceptualRoom(room_code)
24:      create newtarget and add it on destination vector
25:   end if
26: end if
27: return OK
```

entorno poco ruidoso y con una necesidad lumínica de oscuridad. Es posible por lo tanto que haya una serie de preguntas que identifiquen el destino que quiere tomar el usuario y tal vez haya una manera efectiva de ordenar esas preguntas que minimice su número necesario para identificar el destino que se quiere alcanzar. De ser cierta esta hipótesis, tal vez con 2 ó 3 preguntas sobre el estado del usuario el robot fuese capaz de anticiparse a sus necesidades y sugerir un destino.

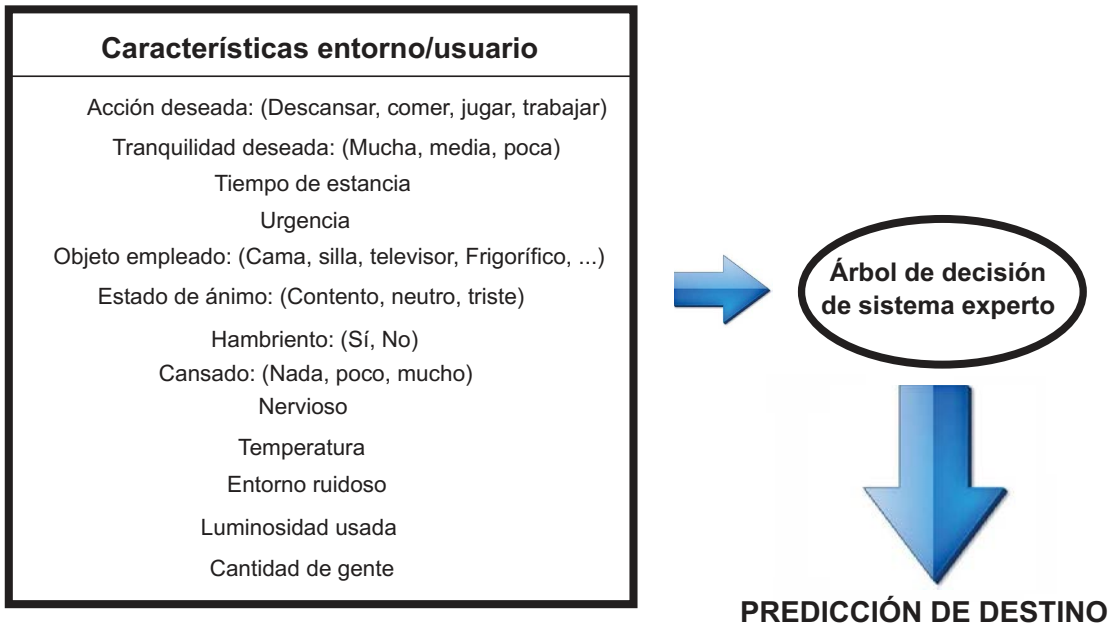


Figura 4.23: Enfoque de deducción de destino orientado a la anticipación de destino en base a experiencias previas.

4.4.1. Variables consideradas

Pese a que potencialmente tiene mucho interés, este enfoque se ha quedado en un estado inicial. Las variables consideradas para construir los árboles de decisión en los que se basa este método son:

- Acción. Es la variable que guarda la acción que quería hacer el usuario cuando inició el desplazamiento. Los valores que se han considerado en las pruebas que

podría tomar esta variable son: Descansar, Comer, Jugar, Leer, Dormir, Beber y Trabajar.

- Tranquilidad Requerida. Es el nivel de tranquilidad que requiere la acción que desea realizar el usuario. Los valores considerados han sido: Mucha, Media y Poca.
- Tiempo Requerido. Es el tiempo que considera el usuario que puede dedicar a la acción deseada, o que tiene disponible para ella. Los valores a tener en cuenta han sido: Mucho, Medio y Poco.
- Urgencia. Es la valoración subjetiva del usuario sobre la urgencia de la acción. Los valores que se han considerado son: Mucha, Media, Poca.
- Objeto. Es el objeto que el usuario utiliza en su destino. Los valores para las pruebas han sido acotados y se han considerado los objetos: Cama, Silla, Consola, Comida, Tele, Libro, Ordenador, Frigorífico, Agua, Sofá y Refresco.
- Ánimo. Es el estado de ánimo que el usuario dice tener cuando inicia el desplazamiento. Los valores son: Contento, Neutro, Triste.
- Hambriento. Esta variable representa si el usuario tiene hambre. Los valores contemplados son: Sí, No.
- Cansado. Es el nivel de cansancio que experimenta el usuario. Se consideran los valores: Nada, Bastante, Mucho.
- Nervioso. Esta variable indica si el usuario está alterado. Sus valores pueden ser: Sí, No.
- Temperatura. Es la temperatura que alcanza el destino. Sus valores pueden ser: Calor, Normal, Frío.
- Entorno Ruidoso. Indica si el usuario ha ido a un lugar ruidoso o no. Los valores posibles son: Sí, No.
- Luminosidad Necesaria. Es el nivel de luminosidad que exige la actividad que el usuario quiere realizar. Los valores son: Alta, Media y Baja.

- Cantidad de Gente. Es el número de personas que hay en el destino elegido. Los valores son: Ninguna, Poca, Mucha.

- Estancia. Es la variable objetivo, el tipo de estancia al que el usuario se ha desplazado, supuestamente en función del resto de variables. Los valores tomados en cuenta para las pruebas han sido: Comedor, Salón, Cocina, Dormitorio, Despacho, Aseo, Terraza.

4.4.2. Implantación con árboles de decisión

Este sistema implica que el robot sea dotado de capacidad de aprendizaje sobre casos conocidos. Para llevar esto a cabo se ha probado con un programa desarrollado en lisp ([124]) que extrae un árbol de decisión a partir de un archivo de texto con la información de varios casos de navegación. El árbol de decisión permite que el robot llegue a la conclusión del destino que quiere alcanzar el objetivo basándose en una serie de preguntas. Un ejemplo de árbol de decisión que ha sido generado durante las pruebas de este sistema se puede ver en la figura 4.24. En ese árbol se aprecia que la primera pregunta es sobre el estado de ánimo del usuario, si éste es *contento* se prevé que el usuario se va a dirigir al salón. Cada elipse representa un valor que debe tomar un atributo por lo que dicho valor es la rama del árbol. En ese ejemplo se ve que el destino puede ser adivinado con 1 pregunta (sobre el estado de ánimo), con 3 preguntas (sobre el estado de ánimo, la temperatura y el tiempo requeridos) o con 4 preguntas (sobre el estado de ánimo, la temperatura y tiempo requeridos y la acción deseada).

El proceso sigue esta secuencia:

1. Inicializar datos. Es una función que parsea un archivo de muestras.

2. Construir reglas. Esta función crea un árbol de decisión a partir del archivo de muestras. Cuando se tiene el árbol de decisión, se extraen reglas que serán utilizadas en un motor de razonamiento. El código en lisp de este paso emplea una función de creación de árboles que que recibe las muestras de entrenamiento, el valor objetivo (en este caso el valor de la estancia) y la lista de los valores posibles que puede tomar cada atributo.

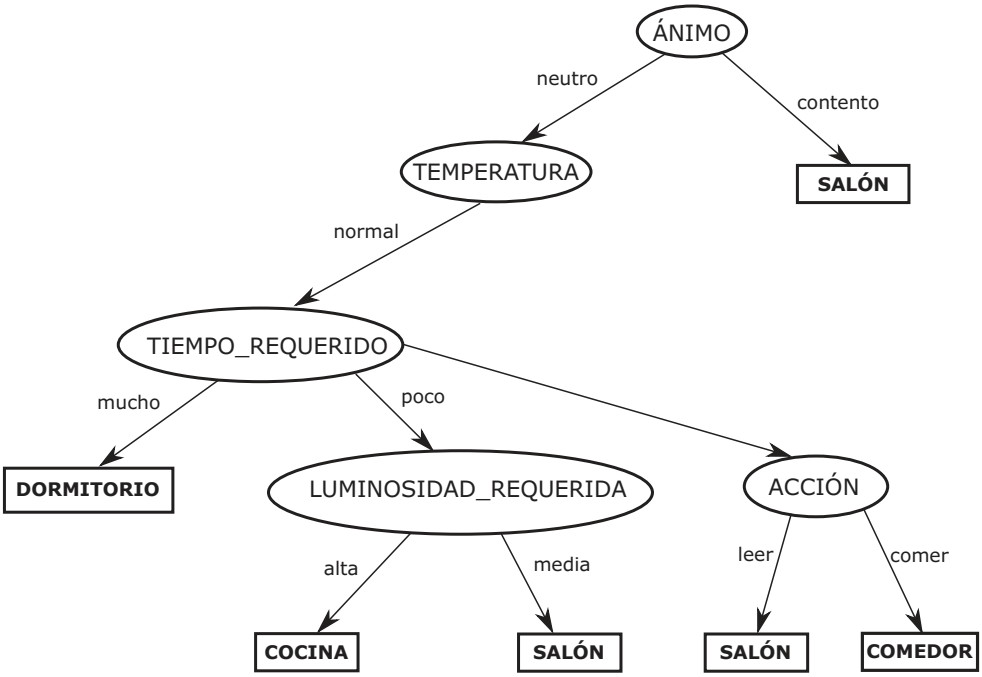


Figura 4.24: Árbol de decisión generado en los experimentos.

```
(defun construye-reglas ()
(setf arbol (create-tree *training.examples*
                 *target.attribute* *attributes*))
(get-rules arbol)
(setf rule-list lista-reglas)
)
```

Una vez que el árbol está creado, se aplica la siguiente función para obtener las reglas del mismo

```
(defun saca-reglas (lista conds-previas)
  (let ((regla))
  (mapcar #'(lambda(x)
       (if  (atom (second x))
                (push (append lista-prueba2
                  (list (remove nil (append (list
                              (list (first lista)
                                    'es (first x)))
                            conds-previas))
                  'THEN
                  (list (list (first (first hypothesis))
                                     (second x)))))
                lista-reglas)
           (saca-reglas (second x)
                  (remove nil (append (list
                                      (list
                                        (first lista)
                                        'es (first x)))
                                conds-previas)))
       )
      )(rest lista))
    )
  )
```

```
    (defun get-rules (arbol)
    (setf lista-reglas nil)
    (set-hypothesis (list (list *target.attribute* '?d)))
    (setf lista-prueba2 (list 'REGLA 'CF 1.0 'IF))
    (saca-reglas arbol nil)
)
```

3. Comprobar el razonador. Con las reglas extraídas en el paso anterior, se invoca un función de razonamiento hacia atrás (backward) que va haciendo preguntas según va unificando las reglas hasta que llega a la conclusión sobre el destino que debe tomar.

La siguiente cuestión a tratar es cómo se han definido las reglas que luego son interpretadas por el motor de razonamiento implementado. Se han definido como un conjunto de reglas formando una lista. Cada una de las reglas tiene una estructura definida donde se diferencian los siguientes elementos:

- Un identificador de regla. Por ejemplo, R1.
- El CF (Factor de certeza) de la regla. Esto es para incluir la posibilidad de que el usuario confirme parcialmente un hecho. Como factor de certeza se admite un valor entre -1 y 1. El -1 significa que existe total certeza para negar el hecho, el 1 significa que existe total certeza para afirmar el hecho. Un valor de cero significa que no se sabe absolutamente nada sobre la veracidad o falsedad del hecho.
- Parte condicional. Empieza con la palabra reservada IF y vienen todos los hechos que deben darse para que se cumpla la condición.
- Parte conclusiva. Empieza con la palaba reservada THEN y le siguen todas las conclusiones que aparecen cuando se cumplen los hechos de la parte condicional.

En la sección de experimentos se adjuntan unas cuantas pruebas hipotéticas. Este enfoque aún no ha sido probado en situaciones reales, todos los ejemplos han sido

hipotéticos. Se propone como trabajo futuro realizar una encuesta o estudio que permita tomar valores reales para probar la eficacia de este sistema.

En este capítulo se han descrito las técnicas implementadas de gestión y acceso de la información semántica. El modelado del entorno ha sido realizado en base a una ontología diseñada para la navegación semántica. Se han detallado los mecanismos de extracción de información implementados con un middleware creado en esta tesis, el cual accede a una base de datos que contiene la ontología. Además, se ha mostrado el proceso de integración con otro sistema de razonamiento basado en Knowrob en el que se ha implementado la misma ontología. Ambas implementaciones son usadas por el planificador para obtener el destino. Por último, se ha presentado en estado preliminar de desarrollo otra alternativa al mecanismo de deducción de destino, basado en árboles de decisión.

CAPÍTULO 5

Adquisición de nuevo conocimiento

"Todo nuestro conocimiento arranca de los sentidos, pasa al entendimiento y termina en la razón." — Immanuel Kant

El sistema de navegación propuesto requiere de un módulo encargado de adquirir nueva información semántica, lo que dota a la navegación de la capacidad de aprender nuevas relaciones entre objetos o entre objetos y estancias. Se considera que esta capacidad es de gran importancia, puesto que el sistema está basado en conceptos subjetivos sobre el entorno que cada usuario puede interpretar de una manera diferente. En las sociedades occidentales se tiende a ignorar todas las posibles interpretaciones y/o matices del entorno que no son tan populares. Somos conscientes de estas diferencias, cuyo origen se debe a que aunque el ser humano modifica su entorno, el entorno también modifica al ser humano y se refleja en su interpretación de las características relevantes del entorno. Así pues, según la cultura o la localización geográfica, se pueden dar grandes diferencias. Por ejemplo, es diferente la organización de la vivienda europea respecto a las viviendas que se pueden ver en otro continente de climatología y características diferentes como África. Los cerca de 1.000 pueblos diferentes que existen en el continente africano hacen difícil hablar de un sólo y único tipo de vivienda tradicional, encontrándose distintos tipos de estancias y organizaciones de

la vivienda para cada pueblo. Es ampliamente conocido que los pueblos esquimales contemplan varios vocablos diferentes para referirse a la "nieve", según variaciones de utilidad/significado (nieve cayendo suavemente o nieve idónea para caminar por ella, por ejemplo). Además, en la cultura japonesa podemos encontrar utensilios domésticos con funciones muy específicas, como por ejemplo almohadones destinados a su uso como asiento en verano, o almohadones para proteger el tatami de marcas. Es por lo tanto un hecho que el ser humano encuentra necesidades de utilidades que cubren distintos objetos que forman su entorno y su vivienda, dependiendo de múltiples factores. Los conceptos de los objetos y la modificación del entorno en función a nuestras necesidades son variables. Por esto se considera importante que un robot destinado a realizar tareas domésticas y actuar en entornos humanos, pueda aprender los matices y la importancia que pueda tener cada objeto para representar una función o asociación con diferentes tipos de estancia. Sin embargo, pese a las diferencias encontradas en la definición y modificación del entorno por los humanos, se mantiene constante la dependencia establecida en capítulos anteriores para modelar el entorno, como si el cerebro humano mantuviera una estructura fija de relaciones objeto-objeto, objeto-utilidad, utilidad-estancia. Y dentro de esta estructura, se pueden añadir las relaciones que se quiera. Se puede hablar de diferentes conceptos de objetos, pero siempre se relacionan con utilidades y con estancias. Es por lo tanto una estructura fija y lo que resta es rellenar con el conocimiento necesario estos conceptos variables, alcanzando el dinamismo que tenemos los humanos a la hora de situarnos y reconocer elementos del entorno.

5.1. Estructura del Sistema de Adquisición de Conocimiento

El sistema de adquisición de conocimiento consta de diferentes módulos para cubrir distintas formas de conseguir que el robot aprenda y relacione conceptos rellenando su base de datos. Conceptualmente hay tres mecanismos diferentes, según la naturaleza de la forma de adquirir información. El robot puede aprender por sí mismo con sistemas de aprendizaje automático a reconocer la semántica del entorno y/o las

Figura 5.1: Maneras de añadir información conceptual al robot.

funcionalidades de los objetos según ciertos parámetros. Esto sería la vía de aprendizaje relacionada con mecanismos de razonamiento. El robot también puede obtener información de la que carezca, consultando en webs orientadas a ofrecer este tipo de información a sistemas de inteligencia artificial, de la misma manera que un ser humano consulta en internet lo que no sabe. Esta es la vía de la consulta web. La tercera vía consiste en lo que el robot puede aprender preguntando directamente a un usuario. Una cuarta vía aunque en realidad no se puede considerar como tal, consiste en la introducción manual de información al sistema.

Por lo tanto, en esta tesis se han contemplado varias maneras de poder añadir información del entorno al modelo, es decir, de que el robot pueda aprender nuevos conceptos. La figura 5.1 recoge estos enfoques que acaban de comentar y se describen a continuación:

Información a priori Es decir, información añadida manualmente. En un primer momento, la información del entorno es introducida por el usuario, accediendo directamente a las tablas de la base de datos. Sin embargo, la información de los lugares y los objetos podría no estar completa, con lo que es necesario que el

robot pueda adquirirla en el futuro. Aunque el usuario siempre tiene la opción de introducir la información por él mismo, el sistema puede deducir conceptos que el usuario no ha proporcionado por inferencia implícita en las relaciones de los conceptos que sí ha introducido.

Obtención de información mediante los sensores del robot Mientras navega, el robot puede detectar objetos del entorno y añadir la información de relaciones con el lugar en el que se encuentra y con otros objetos que lo rodean. El sistema sensorial funciona agregando distintos sistemas de detección de objetos. Algunos de estos sistemas son fruto de trabajos previos en la dirección de Trabajos Fin de Máster y han sido publicados en [6] y, más recientemente, en [55].Estos sistemas funcionan detectando los contornos de los objetos para la detección de objetos genéricos y empleando descriptores para identificar objetos específicos. De este modo, se pretende que el robot tenga la capacidad de reconocer que un objeto es una silla (incluso viendo sillas nuevas por primera vez) pero además, de reconocer si es necesario un objeto concreto (en este caso, ha debido de ser entrenado para reconocerlo). También está previsto poder integrar detectores de muebles como el de [85]. No obstante, debido a las dificultades típicas de la tarea de detección de objetos, el sistema sensorial también incluye un detector de marcas y etiquetas para identificar objetos más difíciles de reconocer por contornos y que tampoco han sido entrenados con el sistema de descriptores. Uno de estos sistemas de detección de etiquetas que ha sido usado está incluido en el paquete de ARToolkit. Los detalles del sistema sensorial se explican en la sección 3.4.2.

El sistema de navegación semántica cuenta con un módulo de exploración que es el responsable de asociar los objetos percibidos o las estancias identificadas con posiciones geométricas o nodos topológicos. Esta manera de adquirir información viene determinada por la habilidad de clasificar una estancia mediante sus sensores (de objetos o de circunstancias como las personas detectadas o el ruido ambiente). Es importante señalar que esta tesis pretende innovar en el ámbito de la clasificación de estancias para sistemas de navegación. El enfoque típico de la manera de clasificar estancias está representado en la figura 5.2, donde la información utilizada viene dada por la apariencia de la estancia y los

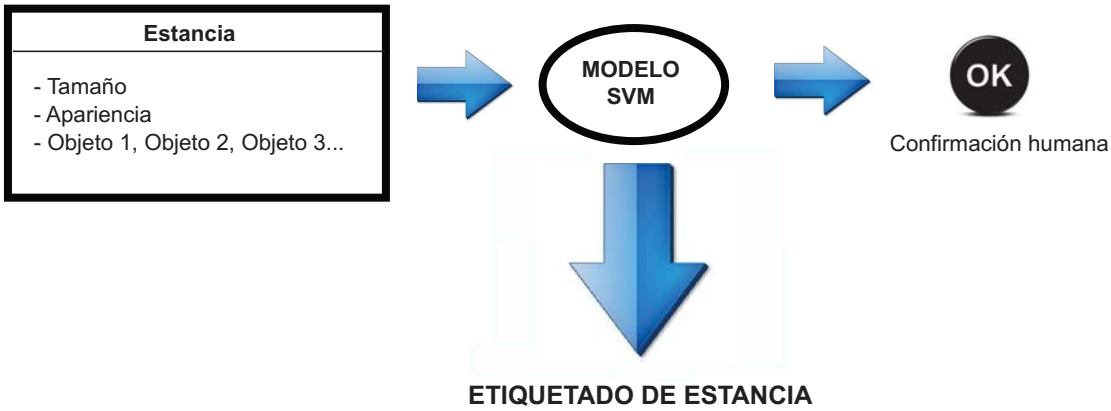


Figura 5.2: Enfoque típico de etiquetado en navegación semántica.

objetos que contiene. En la siguiente sección 5.2 se aborda la clasificación según los objetos detectados pero se incluye un nuevo enfoque en la sección 5.3.

Obtención de información mediante interacción con el usuario Cuando el robot necesita más información, se espera que el robot interactúe con el humano mediante comunicación oral con un sistema de diálogo desarrollado en [84] o por consola de comandos.

Obtención de información usando bases de datos de conocimiento Existen servidores web de acceso público que ofrecen una base de datos de conocimiento a sistemas inteligentes como robots o algún software de Inteligencia Artificial. El robot puede conectarse a estas bases de conocimiento para completar significados, utilidades y otras propiedades de los objetos y lugares del entorno. Esto permite añadir relaciones entre conceptos. En este capítulo se detalla el servidor web utilizado y la implementación de un módulo de consulta en el robot.

Independientemente de la vía empleada para que el robot adquiera nueva información, el módulo que gestione este proceso debe escribir en la base de datos el nuevo conocimiento. En las peticiones siguientes de destino por parte de los usuarios, el robot podrá contar con lo aprendido, puesto que para cualquier planificación de trayectoria que sea requerida el robot siempre tiene en cuenta toda la información que hay en la base de datos.

5.2. Clasificación de estancias mediante objetos detectados

Como se ha mencionado en distintas secciones de esta tesis, la información semántica es necesaria para identificar la estancia a partir de un vector de objetos detectados. Como parte de las funciones del explorador, se contempla un método que tiene este cometido de identificar una estancia en función de sus objetos. Además el explorador (sección 3.3) cuenta con la capacidad de diferenciar objetos nuevos que aparecen en el entorno de objetos ya conocidos, por lo que los objetos que no están siendo observados por primera vez son ignorados en el momento de la clasificación de la estancia. Y para los objetos nuevos, se consulta al razonador por los tipos de estancia diferentes que pueden contener cada objeto. Si se percibe un objeto cuya presencia define la estancia donde está ubicado, la consulta habrá devuelto un único tipo de estancia y eso habrá clasificado la estancia en la que se encuentra. Si la consulta no devuelve una única respuesta, es porque el objeto en cuestión no es lo suficientemente discriminatorio como para poder concluir que se trata de un tipo concreto de estancia.

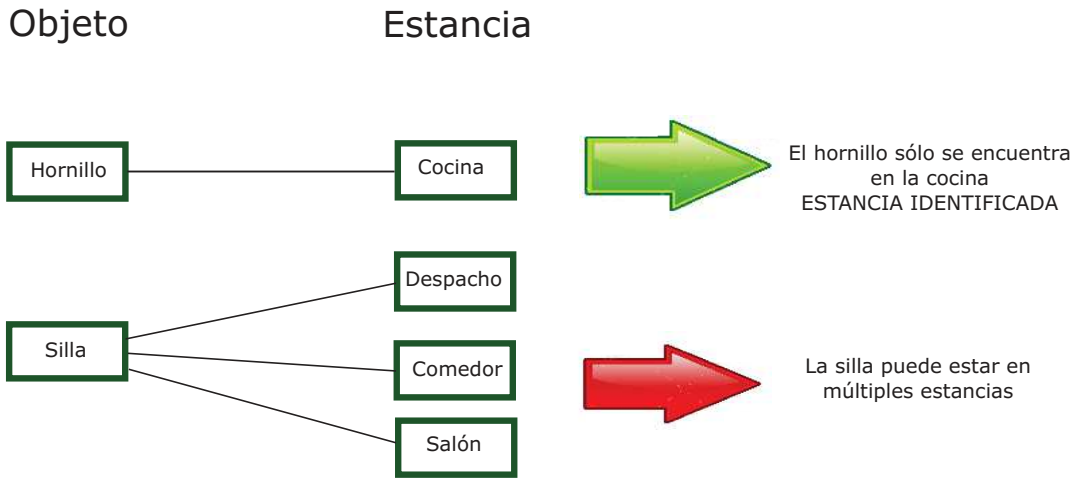


Figura 5.3: Enfoque de clasificación de estancias en función de los objetos contenidos.

El primer enfoque de clasificación es sencillo, se parte de la premisa de que los objetos que sólo están en un determinado tipo de estancia sirven para identificar esa

estancia. Es decir, si el lavabo únicamente está en el cuarto de baño, si veo un lavabo quiere decir que estoy en el cuarto de baño. Esto se representa en la figura 5.3, donde se muestra que un hornillo identifica una cocina pero una silla no es suficiente para identificar la estancia porque puede estar en varias diferentes.

Además, este sistema permite la definición múltiple de tipos de estancias. Si se tiene que el sofá está en el salón, las mesas grandes en el comedor y el hornillo en la cocina, si en una misma estancia se detecta un hornillo, una mesa y un sofá se concluye que esta estancia es un salón-comedor y además cocina.

El funcionamiento de este sistema de clasificación se apoya en una ampliación del enfoque típico de clasificación de estancias. La propuesta es que las estancias son clasificadas en función de lo que se puede hacer en ellas. Es decir, que la cocina es cocina porque se puede cocinar en ese lugar, el dormitorio lo es porque se puede dormir, etc. Esto da sentido a que una misma estancia pueda ser etiquetada de distintas formas, el caso del comedor-salón-cocina resulta correcto porque realmente es una estancia donde se pueden realizar todas las actividades que se supone que deberían poderse realizar. Pero además esto está relacionando estancias con utilidades y sin embargo las utilidades están relacionadas con objetos. Si se puede cocinar en la cocina es porque hay un hornillo, del mismo modo que si en el dormitorio se puede dormir es porque hay una cama. Por esto, en la clasificación de estancias por objetos se puede ampliar a las utilidades.

Este concepto se refleja en la figura 5.4. Y además se amplía de forma natural con *características* y *significados*, siendo la característica relevante porque puede cambiar la utilidad de un objeto (el agua fría no tiene los mismos usos que el agua caliente, por ejemplo) y el significado es el concepto que trata de reflejar que algunas utilidades evocan ciertos sentimientos o afectan de una forma concreta al estado del ánimo del usuario, lo que también se puede emplear para diferenciar un lugar de otro. Además, los objetos con utilidades relacionadas tenderán a estar en el mismo lugar, dando mayor eficiencia a un sistema de navegación semántico que tenga en cuenta utilidades. Este sistema de clasificación de estancias analiza los objetos detectados, con su característica si es conocida, se consulta en la base de datos la utilidad que tiene cada objeto y con esa utilidad se averigua además si existe algún significado subjetivo añadido y en cualquier caso si el objeto sirve para algo concreto y además

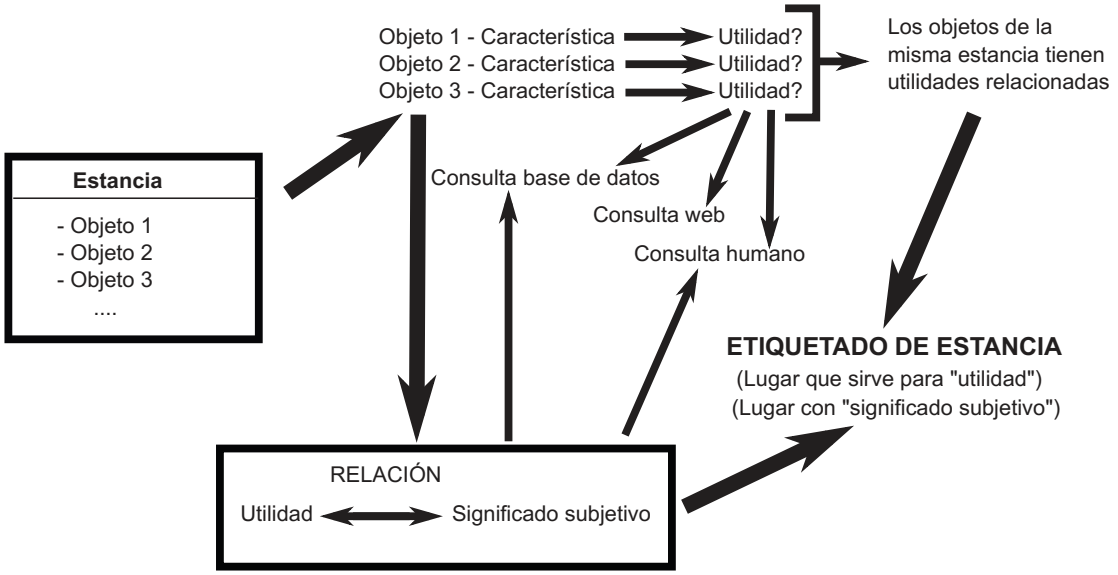


Figura 5.4: Enfoque de clasificación de estancias en función de la utilidad de la estancia.

no se encuentra en ningún otro tipo de estancia, la estancia actual queda identificada, se conoce su utilidad y las emociones o sensaciones que puede transmitir.

Sin embargo puede darse el caso de que en la base de datos no esté la información de la utilidad que tiene el objeto. En este caso, se ha implementado una ayuda al clasificador que consiste en un subsistema que consulta a una web de conocimiento externa (sección 5.4) que por lo tanto es otro medio de obtener información, es decir, de aprendizaje de nuevos conceptos. Y si eso falla, el clasificador solicita al resto del sistema que interaccione con el usuario para preguntarle directamente por la utilidad de dicho objeto. El sistema de interacción con el usuario contempla esta posibilidad y otras más de aprendizaje mediante diálogo humano-robot (sección 5.5).

5.2.1. Descripción del algoritmo

Este clasificador de estancias en base a los objetos y sus utilidades es lanzado por el razonador dentro del módulo explorador cuando este recibe información sobre objetos percibidos de parte del módulo sensorial. El método invocado es *identificarEstanciaFísica*, cuyo pseudocódigo se muestra en el algoritmo 6.

El pseudocódigo corresponde con el método implementado según el razonador construido sobre la base de datos relacional, por eso se muestra la consulta SQL. En la sección 4.2.3 se puede encontrar la consulta lanzada al servidor que interacciona con la ontología basada en KnowRob.

Algorithm 6 identificarEstanciaFísica (Implementación con bases de datos)

Require: $vector < String > listaNombresObjetos$, Estancia* *estancia*
Ensure: void

```
1: semantic_navigation :: consultasrv1;
2: for all elements of listaNombresObjetos do
3:    string consulta = "selectnombre_estanciafrom"+
4:    "estancia_objeto_conceptualwhere"+
5:    "nombre_objeto = " + element ;
6:    srv1.request.nombre = consulta;
7:    if !clientConsulta.call(srv1) then
8:       ROS_ERROR("Failedtocallserviceconsultor");
9:       return
10:   end if
11:   if srv1.response.resultado.size() == 1 then
12:      Identificada estancia, actualizar el valor de su tipo con el resultado
13:      Actualizar la base de datos para guardar la información de la estancia
14:   end if
15:   if srv1.response.resultado.size() > 1 then
16:      Actualizar todas las estancias posibles de estancia
17:   end if
18: end for
19: return
```

Este sistema es sencillo pero efectivo, si una estancia no contiene ningún objeto específico y característico de un tipo de estancia, no se puede saber qué tipo de estancia es. Del mismo modo que una estancia vacía sin ningún objeto en su interior puede ser cualquier cosa. Las probabilidades se van acotando según van apareciendo objetos en la estancia. Aunque estos no sean exclusivos de un tipo de estancia concreto, sí pueden corresponderse a un número limitado de estancias, con lo que ese lugar queda etiquetado como *posiblemente* una de ellas. Cuando aparece un objeto que sólo puede estar en un tipo de estancia según el conocimiento previo, la estancia

queda identificada y esa información se traslada a la base de datos. De manera que queda añadida permanentemente y corresponde a un aprendizaje.

5.3. Clasificación de estancias en función de las acciones de las personas

Estudiando la literatura hasta ahora, queda patente que las características que extraen los investigadores para clasificar un lugar se basan en lo que se puede encontrar en el entorno. Hay múltiples trabajos [157], [92], [130], [81], [147] que consiguen asignar una etiqueta a un lugar por los objetos que el sistema reconoce en él. En esta tesis también se ha aprovechado la relación existente entre objetos y estancias, como se ha mostrado en capítulos anteriores. Un lugar donde se ha detectado un televisor, es etiquetado como *sala de estar*. Aparte de emplear la información que proporcionan los objetos sobre el lugar en el que se encuentran, otros autores han profundizado más a la hora de ampliar el abanico de características que directa o indirectamente proporcionan una información semántica diferenciadora que permita identificar un lugar. En el desarrollo de esta tesis se han visto trabajos que buscaban destacar sobre otros mediante la inclusión de todas las características posibles del entorno que pudieran aportar más conocimiento para etiquetar semánticamente una estancia [101]. Así pues, se han explotado características como la forma geométrica de la estancia, los objetos que hay en ella, etc. Estos trabajos han servido como motivación para continuar tratando de ampliar los puntos de vista que pueden servir para clasificar una estancia.

Para poder aprovechar todas las fuentes de información semántica que puede captar el robot, se ha tenido también en cuenta la información que transmite lo que están haciendo las personas que están en ese entorno. La idea se basa en las mismas premisas sobre las que se asienta la navegación semántica: el ser humano clasifica los lugares en función de a lo que los quiere destinar, es decir, de las acciones que va a realizar en dichos lugares. Sin embargo ahora en vez de centrar la atención en las características estáticas del entorno, se centra la atención en lo que sucede en dicho entorno cuando hay personas actuando sobre él. Esto plantea un nuevo reto, que consiste en poder dotar al sistema de la capacidad de discernir lo que sucede en un lugar, para luego

poder etiquetar ese lugar en función de lo que estuviera sucediendo. Por ejemplo, las personas que están en una biblioteca mantienen silencio y la cantidad de movimiento por persona es bajo, si un robot entra en un lugar donde apenas aprecia movimiento ni ruido, puede que sea una biblioteca o lugar que exija tranquilidad.

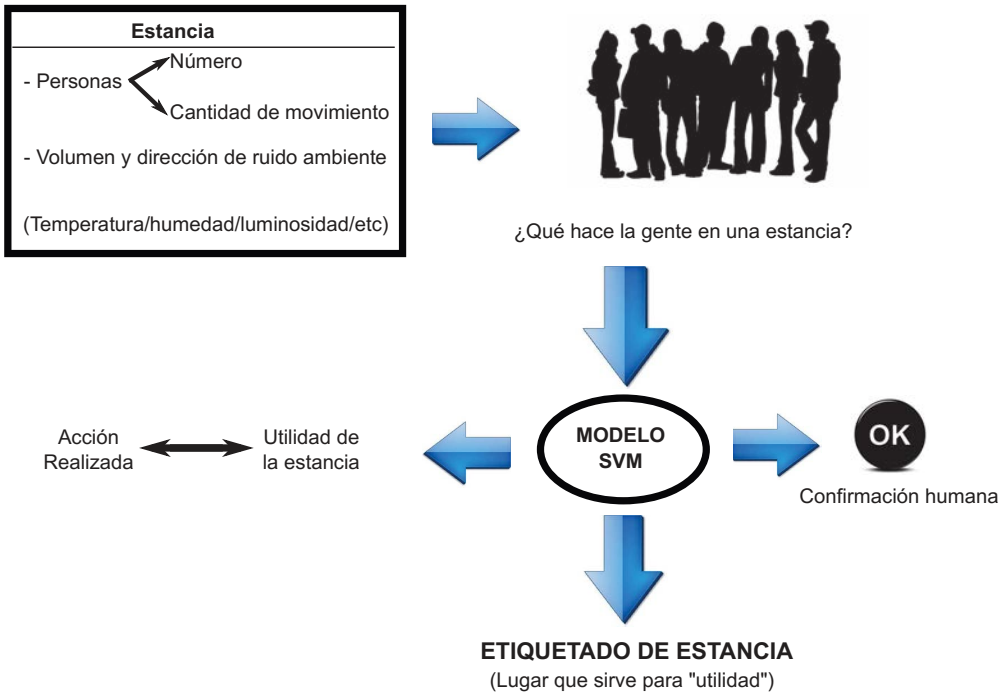


Figura 5.5: Enfoque de clasificación de estancias basado en la utilidad de la estancia, deducida en función de lo que hacen las personas en ella.

La elección de las características que puede interpretar el robot para este objetivo ha sido hecha cuidadosamente. En una primera aproximación al problema, se ha tenido en cuenta información visual para detección de personas e información sonora para medir el volumen del ruido ambiente y su dirección. Esto amplía el enfoque descrito en la figura 5.5.

Es un enfoque diferente donde se pretende etiquetar el entorno en función de lo que la gente hace en dicho lugar. La idea es que dependiendo de la actividad que tengan las personas en una habitación, se puede intentar deducir qué tipo de habitación es, lo que puede dar lugar a la identificación de distintos lugares como muestra la Figura 5.6. Averiguar exactamente qué es lo que están haciendo un grupo de personas en

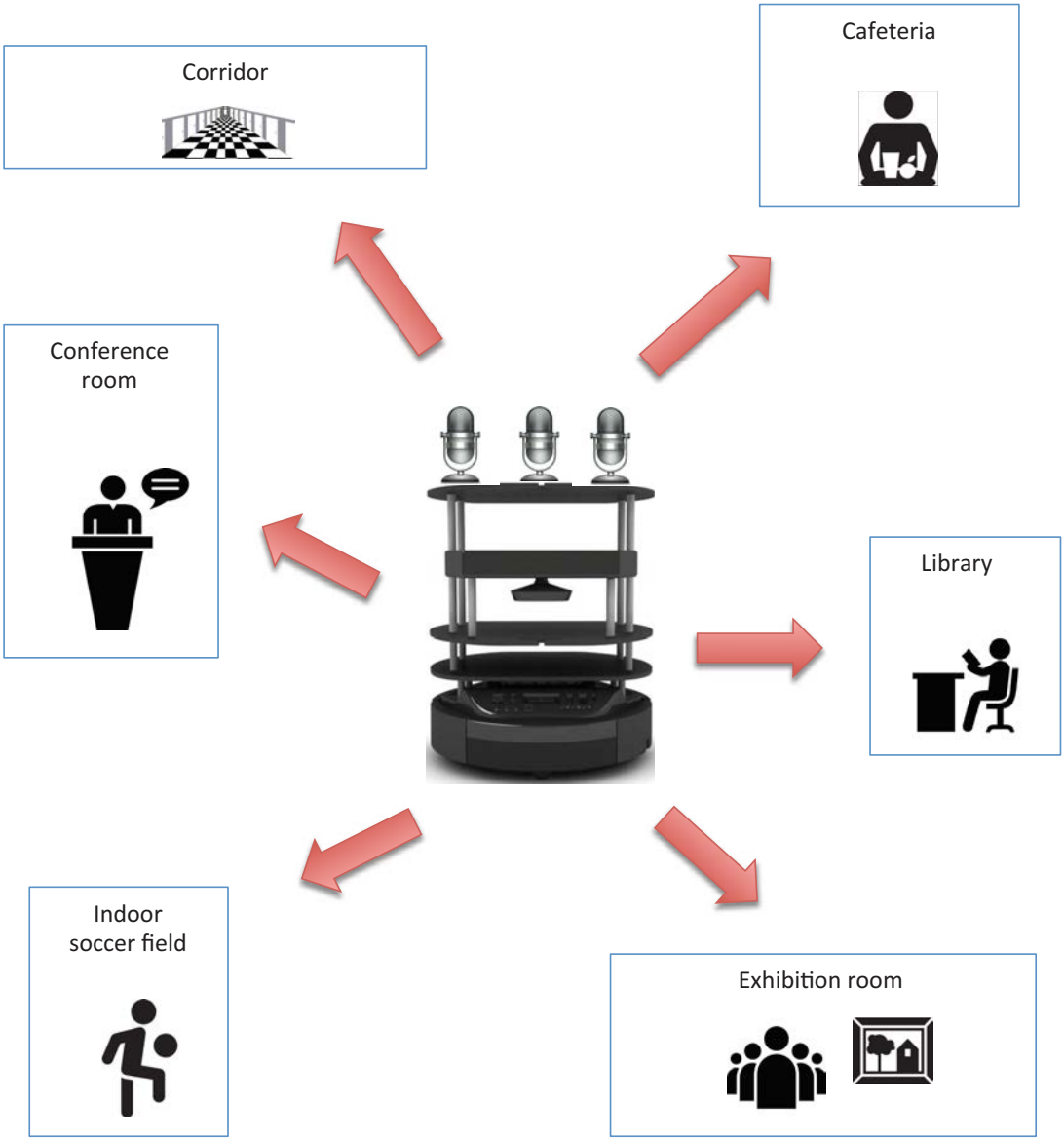


Figura 5.6: Detección de diferentes escenarios en función de lo que hace la gente en ellos.

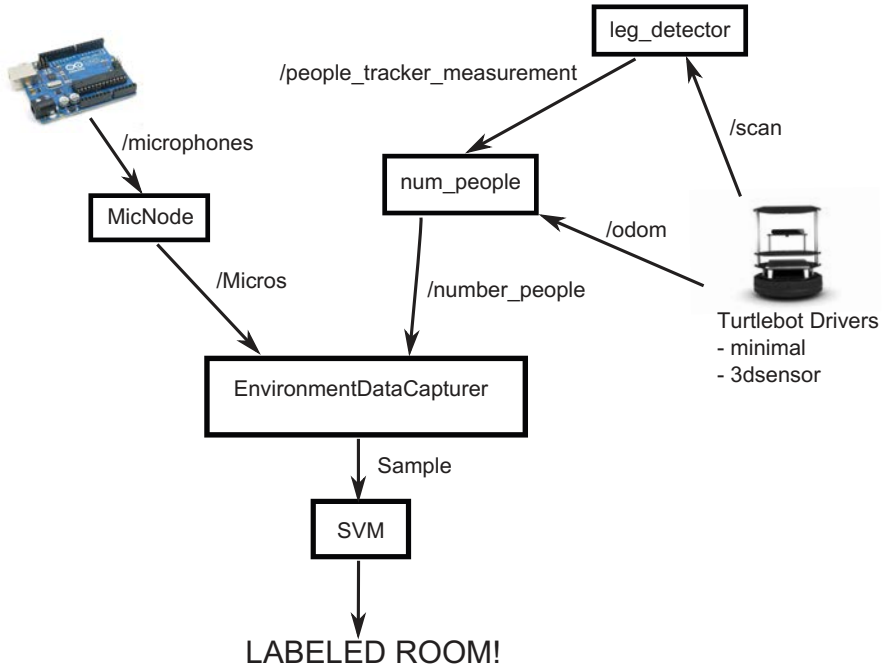


Figura 5.7: Diagrama del sistema completo.

una habitación puede ser complicado para un robot. En esta tesis el sistema ha sido enfocado primero en diferenciar de forma genérica algunos detalles de las personas que cambian según lo que se esté haciendo. Se ha prestado atención al número de personas, en lo que se mueven y en el sonido ambiental. Fijándonos únicamente en ello, podemos saber que en determinadas localizaciones se están realizando acciones diferentes y, por lo tanto, etiquetar esas localizaciones.

Los módulos y elementos que componen este sistema de etiquetado, incorporado en un robot *Turtlebot 2*, se muestran en la figura 5.7. Estos son:

- Los drivers relativos a mantener operativo el robot móvil y que proporcionan información odométrica, además de los relativos a la funcionalidad de la cámara RGB-D. En el caso del Turtlebot 2, estos drivers son *minimal* y *3dsensor*.

- El nodo *leg_detector*, obtenido de la web de ROS (ver sección 5.3.1) y que se encarga de la detección de personas. Es software libre.

- El nodo *num_people*. Es el encargado de obtener la información de las personas detectadas y sus desplazamientos en un intervalo determinado. El muestreo se realiza cuando el robot está inmóvil para que el propio movimiento del robot no altere la muestra. Si el robot está en movimiento, se inhibe el muestreo. Otra opción hubiera sido tener en cuenta y cancelar el movimiento del robot, pero se desestimó para que el código fuera más simple y ligero.
- Conjunto de micrófonos y arduino. El arduino de la figura recibe los datos de tres micrófonos. Para recoger el sonido ambiental, se ha diseñado una estructura que contiene los micrófonos y se acopla al turtlebot. El arduino transmite la información de los micrófonos mediante un mensaje en el topic */microphone*s.
- El nodo *MicNode*. Este nodo muestrea la información que recibe de los micrófonos y envía cada muestra en un mensaje en el topic */Micros*.
- El nodo *EnvironmentDataCapturer*. Este nodo recibe las muestras de los datos del sonido y del movimiento de las personas y se encarga de fusionar ambos datos en una muestra síncrona. Esta muestra es guardada en un fichero para entrenar la Máquina de Soporte Vectorial o puede ser enviada a una Máquina entrenada para clasificar la estancia.
- SVM. Es el módulo que gestiona la Máquina de Soporte Vectorial.

El sistema muestrea cuando el robot está quieto. Para tomar muestras, se puede teleoperar el robot alrededor de una habitación, dejándole quieto en algunas posiciones para permitir el muestreo. El robot también cuenta con un nodo wandering que permite realizar el muestreo de forma autónoma, haciendo deambular al robot y deteniéndolo tras un tiempo determinado durante un intervalo también determinado. Además, se puede dejar al robot detenido orientado hacia la zona visual más espaciosa.

5.3.1. Detección de personas

Puesto que este sistema de etiquetado semántico necesita identificar lo que está haciendo la gente, el primer tema a tratar es incluir un sistema de detección de personas. Se ha usado un método disponible en la web oficial de ROS [1]. Este algoritmo

[1] http://wiki.ros.org/leg_detector

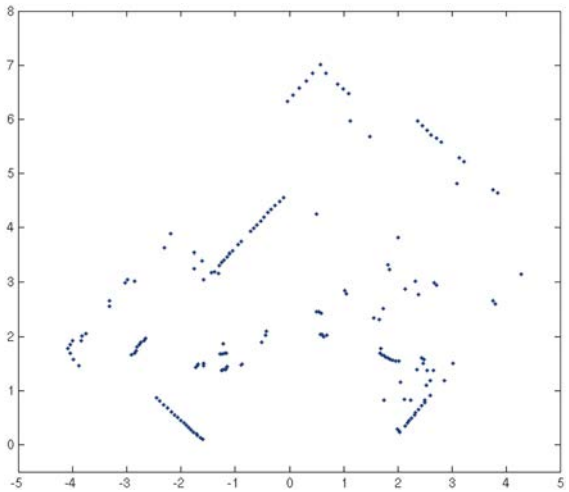


Figura 5.8: Ejemplo de los datos del láser en una oficina típica mostrado en [4].

se basa en la detección de piernas para deducir que se está percibiendo una persona. El encargado del mantenimiento de este método lo comenta brevemente en [78]. Este autor también necesita un sistema de detección de personas y reduce el problema a detectar piernas. Su técnica de detección de piernas se basa en el algoritmo de Arras de [4] y emplea una implementación desarrollada en Willow Garage por Caroline Pantofaru (más información en la misma referencia). Todos ellos usan un grupo de clasificadores de bajo nivel para estimar la probabilidad de que el escaneo láser recibido se corresponda a la lectura de una pierna. El siguiente paso es analizar las probabilidades de que la lectura corresponda a una pierna fijándose en restricciones de distancia de unas piernas con otras. Un algoritmo empareja las piernas individuales que corresponden a una persona bajo estas suposiciones y realiza el seguimiento del par de piernas resultante. Así pues, el algoritmo de detección de piernas identifica dónde está la gente en una estancia usando sólo la información del láser, como muestra la figura 5.8

El nodo *num_people* se subscribe a los topics *people_tracking_measurements* y *odom*, tal como se muestra en la Figura 5.7. El propósito de este nodo es publicar el número de personas que hay en una estancia en un momento determinado. Además también publica datos sobre la cantidad de movimiento de dichas personas.

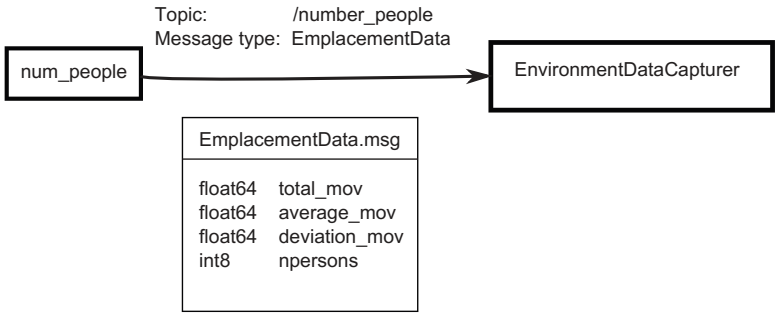


Figura 5.9: Estructura del tipo de mensaje que contiene los datos de la ocupación de un entorno. Se publica en el topic *number_people*.

Recibe información del nodo *leg_detector* en forma de un vector con todas las personas detectadas en ese momento. Este vector contiene la información del identificador de cada persona, el grado de fiabilidad de la detección (que indica lo seguro que está el sistema de que las piernas detectadas son realmente piernas) y su posición actual. También necesita recibir la información de la odometría mediante el topic *odom* para calcular la posición actual y velocidad actual del robot. De este modo, teniendo las posiciones relativas de las personas respecto al robot y la posición absoluta del robot respecto al entorno, se obtienen las posiciones de las personas en el entorno.

El nodo funciona recopilando todos los datos recibidos durante un intervalo de tiempo predefinido, teniendo en cuenta sólo los datos recibidos cuando el robot se encuentra inmóvil. En la duración del intervalo, se forma un vector con objetos del tipo *PersonaDetectada* que almacena a todas las personas detectadas durante dicho intervalo. Cada persona es diferenciada con un identificador y cuando se detecta la misma persona dentro del mismo intervalo, se actualiza la cantidad de movimiento de esa persona. Esto se calcula conociendo la posición actual y la posición anterior de la persona y el tiempo de muestreo. Cuando el intervalo finaliza, se procesa la información almacenada. El vector contiene toda la información relativa al movimiento de las personas que estaban presentes en el escenario. Se añade la media del movimiento por persona y la desviación típica. Esta información se publica con un mensaje del tipo *DatoOcupación* (Figura 5.9) en el topic *number_people*.

$$\forall p_i, dp_i = \sum_{t=0}^{MAX\_TIME} |PX_{new} - PX_{old}| + |PY_{new} - PY_{old}| \tag{5.1}$$

El desplazamiento detectado por el nodo *num_people* tiene en cuenta el movimiento realizado en las dos dimensiones del suelo. Una persona p_i es detectada en el intervalo de tiempo t y la persona se ha movido una distancia dp_i. Esto se representa en la ecuación 5.1. El movimiento de una persona en una muestra es la diferencia en valor absoluto entre la posición actual en el eje X (PX_{new}) y la posición anterior (PX_{old}) y a esto se le suma la diferencia en valor absoluto entre la posición actual en el eje Y (PY_{new}) y la posición anterior (PY_{old}).

$$D_t = \sum_{i=1}^{Np} dp_i \tag{5.2}$$

Las muestras son tomadas en un intervalo de tiempo configurable. Los experimentos llevados a cabo y fueron realizados con un intervalo de 3.5 segundos. El desplazamiento total D_t es la suma de los desplazamientos de todas las personas identificadas Np en ese intervalo de tiempo (Ecuación 5.2).

$$\bar{x} = \frac{D_t}{Np} \tag{5.3}$$

El cálculo de la media aritmática se obtiene para conseguir la media del desplazamiento por persona en la muestra (Ecuación 5.3) y la desviación típica (Ecuación 5.4) se añade para proporcionar más información a la muestra.

$$\sigma = \frac{\sum_{i=1}^{Np}(dp_i - \bar{x})^2}{\bar{x}} \tag{5.4}$$

El nodo *num_people* también permite modificar el coeficiente de certeza para identificar a una persona como tal. En los experimentos este coeficiente se ajustó a un valor de 0.7.

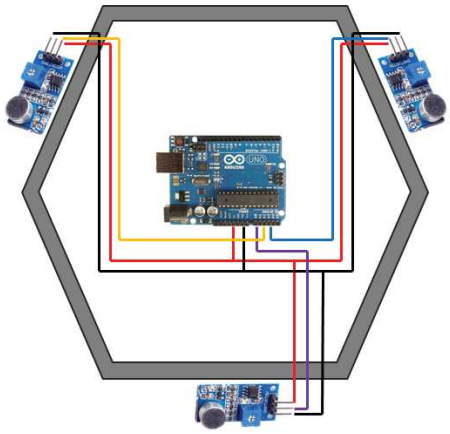


Figura 5.10: Diagrama esquemático de la estructura de micrófonos.

5.3.2. Adquisición de datos del sonido ambiente

Se ha incluído un conjunto de micrófonos a este sistema. Toda la información sonora recibida es captada por un dispositivo formado por tres micrófonos y un microcontrolador Arduino UNO para procesar los datos adquiridos mediante los micrófonos (Figura 5.10). Este dispositivo está alojado en una estructura con forma de anillo hexagonal que se acopla en la parte superior del Turtlebot. La estructura de anillo se ha obtenido con una impresora 3D y está diseñada para orientar los micrófonos en direcciones distintas para poder estimar la zona más ruidosa. Partiendo de estas características, el sistema es capaz de aprender sobre la situación acústica del entorno.

El nodo *NodoMic* recibe la información del topic *microphones*, la cual es enviada por el arduino. Los micrófonos muestrean el sonido ambiente con otro intervalo de tiempo predefinido. Se ha ajustado para coincidir con el tiempo de muestreo de personas. El nodo procesa la información y la publica en el topic *Micros*.

5.3.3. Fusión multimodal de la información

El nodo *CapturadorDeDatosDelEntorno* es el responsable de recopilar toda la información sobre el entorno y gestionar las muestras para etiquetar la estancia. Por lo tanto, este nodo está suscrito a todos los topics que pueden dar información sobre lo que están haciendo las personas. Recibe información de los topics *number_people*

y *Micros*. El primer paso para obtener una muestra fiable, es asegurarse de que toda la información es concurrente. La recepción de los mensajes en topics es asíncrona, esto implica que hay que controlar la recepción de datos. Hay definida una constante *MAX_TIME* para la duración de la ventana temporal. Si los datos recibidos de los dos topics (sonido y dato de personas) llegan al nodo con una diferencia de tiempo menor que esta ventana temporal, entonces los datos son considerados concurrentes. Por otro lado, otra constante definida como *TIEMPO_ENTRE_MUESTRAS*, indica por cuánto tiempo el sistema debe esperar antes de volver a procesar una muestra.

Este nodo realiza la fusión de los datos recibidos y con ello se consiguen las muestras para la clasificación del escenario. Primero, estas muestras se escriben en un fichero al que luego accede la Máquina de Soporte Vectorial (SVM) para el proceso de entrenamiento. Una vez que la SVM ha sido entrenada, se pueden clasificar nuevas muestras y etiquetar las estancias.

La estructura del vector de características que almacena cada muestra aparece en la ecuación 5.5. M_1 es el dato del micrófono 1, M_2 es el dato del micrófono 2, M_3 es el dato del micrófono 3, N_ps es el número de personas en la muestra, D_t es el desplazamiento total de las personas, $\bar{x}$ es la media aritmética del desplazamiento y σ es la desviación típica.

$$Vector\_características = \{M_1, M_2, M_3, N_ps, D_t, \bar{x}, \sigma\} \quad (5.5)$$

5.3.4. Entrenamiento de la Máquina de Soporte Vectorial

Se ha desarrollado un programa con una implementación de una Máquina de Soporte Vectorial (SVM). La SVM se ha tomado de una librería de código abierto de funciones principalmente orientadas a la visión artificial en tiempo real. Esta librería se llama OpenCV. La implementación de SVM ofrecida por OpenCV ha sido ampliamente utilizada en algunos trabajos (especialmente lo que tratan de problemas de visión artificial) como [89][69].

Se han ajustado los parámetros de la SVM. La librería utilizada permite configurarlos. Los parámetros son:

- **svm_type**. Es el tipo de la formulación de la SVM. El valor establecido es CvSVM::C_SVC. Esta opción es para clasificadores que contemplan múltiples

clases y permite una separación imperfecta de clases con una penalización para valores atípicos.

- **kernel_type**: Es el tipo del núcleo de la SVM. El valor seleccionado es CvSVM::LINEAR. Esta configuración es la más rápida. No se realiza ningún mapeo, la discriminación lineal se realiza en el especio de características original.
- **term_crit**. Este es el criterio de finalización del las iteraciones del proceso de entrenamiento de la SVM que resuelve un caso parcial del problema de la optimización cuadrática restringida. La tolerancia y el máximo número de iteraciones también son establecidas. En este sistema, el criterio de terminación es CV_TERMCRIT_ITER, lo que quiere decir que el algoritmo siempre termina después un número definido de iteraciones. Ese número ha sido establecido en 700 iteraciones como máximo.

Cada muestra de entrenamiento para el algoritmo de la SVM se compone de una observación D_i y su clasificación C_i. El conjunto de entrenamiento es representado por la ecuación 5.6, donde Υ es el conjunto de clases. En este sistema se supone que las clases de las muestras de entrenamiento son conocidas previamente. El objetivo es entrenar un sistema de clasificación para ser capaces de generalizar a partir de ese conjunto de entrenamiento y posteriormente clasificar lugares inexplorados en el mismo o en distintos entornos.

$$S = \{(D_i, C_i) : C_i \in \Upsilon = \{Library, Cafeteria, ...\}\} \tag{5.6}$$

Cuando se ejecuta el programa, se solicita al usuario que introduzca el nombre del archivo de muestras. El archivo es posteriormente generado por el nodo *CapturadorDeDatosDelEntorno*. En la ejecución diferida (usada en los experimentos), el programa también solicita que sea introducido el porcentaje de muestras que van a ser reservadas para el conjunto de entrenamiento de la SVM. Las muestras restantes forman el conjunto de test. Cada muestra tiene una probabilidad de pertenecer a uno de los conjuntos que viene determinada por el porcentaje introducido. Esto hace que diferentes ejecuciones del programa sobre el mismo archivo de muestras produzcan distintos resultados. En los experimentos se realizan varias iteraciones sobre el mismo archivo de muestras.

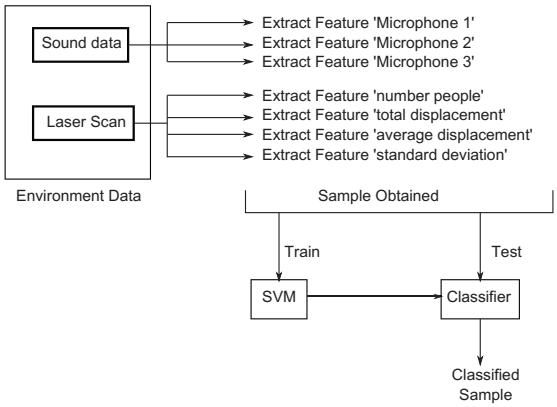


Figura 5.11: Generación, entrenamiento y clasificación de muestras.

El archivo recibido ha sido construido a partir de las muestras tomadas. Estas muestras del entorno son percibidas por los sensores del robot y la SVM es entrenada con ellas. Cuando finaliza el proceso de entrenamiento, se puede clasificar cualquier muestra. Este proceso se ilustra en la figura 5.11. El archivo que recibe la SVM tiene el siguiente formato para cada muestra:

- Identificador de estancia. Se tiene conocimiento del tipo de estancia donde se están tomando muestras. El aprendizaje es supervisado. El primer elemento es la clasificación que conocemos de la estancia.
- Dato de micrófono 1. El sistema de sonido consta de tres micrófonos. El micrófono 1 es el que está situado en el frente del robot.
- Dato de micrófono 2. Este dato corresponde al micrófono que está situado en la parte trasera derecha.
- Dato de micrófono 3. Este es el dato del micrófono situado en la parte trasera izquierda.
- Número de personas. Este dato se corresponde con la cantidad total de personas detectadas en esa muestra.
- Movimiento total. Este dato es la suma del movimiento de todas las personas. Se obtiene, por tanto, una medida del movimiento registrado en la habitación.

- Movimiento medio. Es la media aritmética del movimiento total entre el número de personas. Es una estimación del movimiento medio.
- Desviación típica. Es la desviación típica de la cantidad de movimiento de todas las personas de la muestra.

5.4. Consulta web de conocimiento

En esta sección se comenta uno de los métodos que se han incluido en esta tesis para dotar de mayor autonomía al robot cuando se presenta una situación en la que el sistema no contiene suficiente información para poder deducir el destino. En resumen se trata de que el robot pueda consultar en internet información sobre la utilidad de algún objeto. Esto se ha considerado especialmente útil cuando el usuario solicita ir a un lugar que se define por las acciones que se realizan allí. En este caso es fundamental que el sistema conozca bien las relaciones entre los objetos del entorno y su utilidad. Por ejemplo, si el usuario quiere ir a un sitio donde pueda trabajar y el sistema conoce la ubicación de un ordenador, necesita conocer también que el ordenador sirve para trabajar y de este modo concluir que debe conducir al usuario al lugar donde se encuentra ese objeto.

El conocimiento de las utilidades que tiene un objeto puede ser obtenido mediante consultas en internet. Si se pudiera consultar de forma autónoma, el robot no necesitaría preguntárselo al usuario. Esta es la premisa que ha motivado implementar este sistema y a integrar ConcepNet.

ConcepNet es una red de conocimiento semántico que contiene información sobre conceptos que un sistema de inteligencia artificial debería conocer sobre el mundo. Aunque inicialmente está pensado para que permita entender texto escrito por personas, ha resultado útil para el objetivo descrito en esta sección. En trabajos como [75] detallan en mayor profundidad el funcionamiento de ConcepNet. Está constituido por nodos que representan palabras o frases cortas en lenguaje natural y las relaciones entre ellas. Estos nodos pueden ser entendidos como *conceptos* según los propios autores, aunque aclaran que en realidad son *términos*. En la práctica, tenemos un base de conocimiento semántico de conceptos y relaciones entre ellos. ConcepNet contiene conocimiento básico cotidiano. Por ejemplo, en la figura 5.12 se muestra lo

Figura 5.12: Ejemplo de información que ofrece conceptnet para el objeto SILLA.

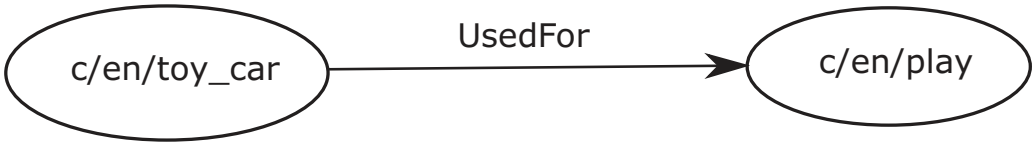


Figura 5.13: Relación entre el concepto *coche de juguete* y *jugar*.

que devuelve la web de ConcepNet cuando se pregunta por el objeto silla (Chair). La imagen muestra que hay mucho conocimiento interesante sobre ese concepto, como que la silla está hecha de madera, que puede estar en una oficina, que debería ser cómoda, que puede estar en un cubículo, etc. Y además está la información que se pretende conseguir de este sistema: su utilidad, pues también se incluye que la silla sirve para sentarse.

5.4.1. Integración de ConceptNet en el sistema

La base de conocimiento de ConceptNet es demasiado genérica y su uso ha sido acotado al caso de aportar una via autónoma al sistema de conseguir la información relativa a la utilidad de los objetos. Se emplea por lo tanto en el momento en el que el robot ha recibido información sobre una acción y pretende saber qué objetos sirven para esa acción. Así pues, primero pregunta a ConceptNet y si no consigue la información que deseaba, pregunta al usuario.

http://api.conceptnet.io/search?**rel**=/r/**UsedFor**&**end**=/c/en/**play**

start :

Coche de Juguete
Pelota
Muñeca
Piezas Ajedrez
Figura de Acción
Cuerda de Saltar
Yoyo
Campo de Baseball

Figura 5.14: Ejemplo de información que ofrece conceptnet para la acción JUGAR.

Se ha implementado un servicio ROS en java. Este servidor atiende peticiones definidas por un tipo de dato con los siguientes campos:

- Relación. Este campo es el tipo de relación que ConceptNet contempla entre los dos conceptos. Para diferenciarlo, al primer concepto se le ha denominado *Palabra* y al segundo *Término*. ConceptNet reconoce varios tipos de relaciones, pero este sistema únicamente se ha interesado por la relación *UsedFor*, que establece que la Palabra es utilizada con el propósito definido por el Término. Por ejemplo, la Palabra *Coche de Juguete* se relaciona con el Término *Jugar* mediante la relación *UsedFor*, como muestra la Figura 5.13.

- Palabra. Este campo es el primer concepto de la relación.

- Término. Este campo es el segundo concepto de la relación.

Entonces, cuando el servidor recibe una petición de consulta a ConceptNet, construye una url con la estructura que se comenta a continuación y el resultado de la consulta web lo devuelve en una lista con los conceptos solicitados. Si se pregunta por

la acción JUGAR, el resultado es el de la figura 5.14, donde se muestran algunos de los conceptos que se devuelven y la construcción de la url de la consulta.

La consulta que realiza el servidor ROS se basa en el intercambio de datos que permiten los objetos JSON. El servidor construye una URL con la información de la consulta que se quiere realizar y devuelve la lista de objetos deseados. Esta URL en el caso utilizado (la obtención de objetos con una utilidad determinada) tiene la siguiente estructura:

URL_BASE/RELACIÓN/ACCIÓN.

Donde se tiene:

- URL base. Al comienzo del desarrollo de esta parte de la tesis, la url base era:

  http://conceptnet5.media.mit.edu/data/5.2/search?rel=/r/

  Sin embargo parece que se ha actualizado y en el momento de la escritura de esta tesis ha funcionado con la siguiente url base:

  http://api.conceptnet.io/search?rel=/r/

- Relación. La relación es la de utilidad, se utiliza la palabra reservada **UsedFor**.

- Acción. La acción es el segundo concepto de la relación (el Término), puesto que se pretende obtener el primer concepto (la Palabra). Se añade

  &end=/c/en/NOMBRE_ACCIÓN.

  La palabra reservada *end* aparece porque es la manera de indicar que el concepto que se está facilitando es el segundo concepto. La palabra reservada *start* se refiere al primer concepto. La figura 5.14 también muestra un ejemplo de construcción de una URL donde se aprecia que se obtienen los primeros conceptos de la relación que concuerdan con el segundo concepto, es decir, los objetos que concuerdan con la acción JUGAR.

Una vez que se obtiene la lista de objetos, el sistema filtra los resultados y se queda con los objetos que conoce en el entorno. De este modo es posible que el robot adquiera la información sobre la utilidad de un objeto sin que haya tenido que decírselo el usuario y a pesar de que no tiene implementado ningún mecanismo de deducción de utilidades. Si esto no sucede, el siguiente paso es el de preguntar al usuario por los objetos que permiten dicha acción mediante un diálogo.

5.5. Adquisición de conocimiento mediante diálogo con humanos

La navegación semántica es más eficiente cuanto más conocimiento posee el robot y por ello se ha dotado al sistema de diferentes mecanismos de adquisición de conocimiento del entorno, que combinados entre sí permiten deducir una considerable cantidad de información de forma autónoma. Sin embargo, cada mecanismo tiene ciertas limitaciones de distinta naturaleza. La capacidad sensorial del robot puede afectar al reconocimiento de elementos del entorno y las consultas web pueden fallar en algunas ocasiones. Es por ello que se considera importante aprovechar cualquier mecanismo de adquisición de conocimiento y el mecanismo más natural, efectivo y práctico desde un punto de vista humano es la consulta directa a un usuario.

En la figura 5.15 se muestran los temas de las preguntas (en verde) que emite el sistema con el objetivo de aprender del usuario. El diálogo con fines de aprendizaje tiene tres objetivos:

- Aprender los objetos que sirven para una determinada acción. Por ejemplo, objetos que sirven para *jugar*. Además el sistema pregunta el grado de lo que se ajusta ese objeto a esa acción. Por ejemplo, para la acción *descansar*, los objetos que dijera el usuario podrían ser silla, sofá y cama. Pero no todos los objetos se ajustan igual de bien a la acción *descansar*. Por eso, el sistema pide una valoración al usuario respecto lo bien que se ajusta cada uno de esos objetos a la acción.

- Aprender las acciones que pueden tener asociado un significado añadido. Por ejemplo, las acciones que son *divertidas* (leer, bailar, ver la televisión, etc).

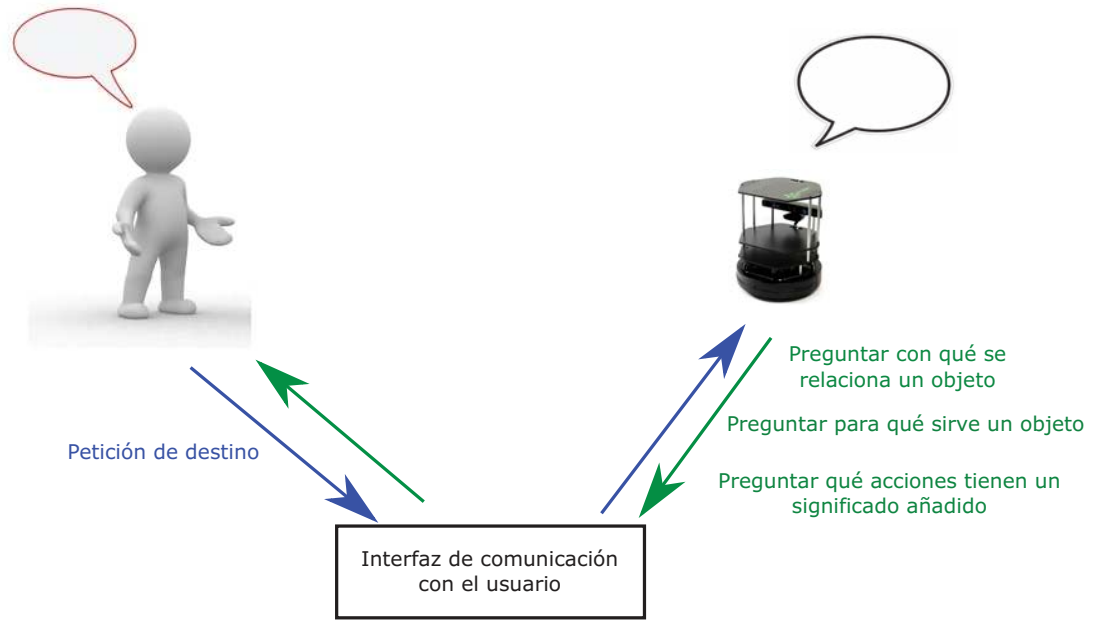


Figura 5.15: Diálogo del sistema con el usuario.

- Aprender las relaciones que tienen los objetos con otros objetos del entorno. El sistema pregunta para cada objeto mencionado por el usuario en el proceso de aprendizaje por sus relaciones con otros objetos. Estas relaciones pueden ser cuando un objeto contiene a otro o cuando un objeto es usado por otro. El diálogo está orientado a obtener esa información por parte del usuario.

La interacción con el usuario consiste en el intercambio de preguntas y respuestas que se van encadenando en función de la información que falta y el robot necesita. La figura 5.16 muestra un ejemplo de interacción con el objetivo de aprender objetos que sirven para una utilidad concreta y el objeto introducido sirve sin ninguna característica especial y encaja muy bien en esa utilidad. Si el objeto necesitara de una característica especial para cumplir esa utilidad, el usuario respondería que sí la necesita y el sistema le pediría que dijera cuál. Por ejemplo, si la utilidad fuese *jugar*, el usuario podría responder que un libro sirve para jugar. Pero no todos los libros, sólo los libros que sean un *manual de un juego de rol*. El usuario debería introducir la característica *manual de juego de rol* que cambiaría la utilidad estándar del objeto

libro. En la figura 5.17 se muestra un ejemplo de un diálogo que permite el aprendizaje de relaciones entre objetos.

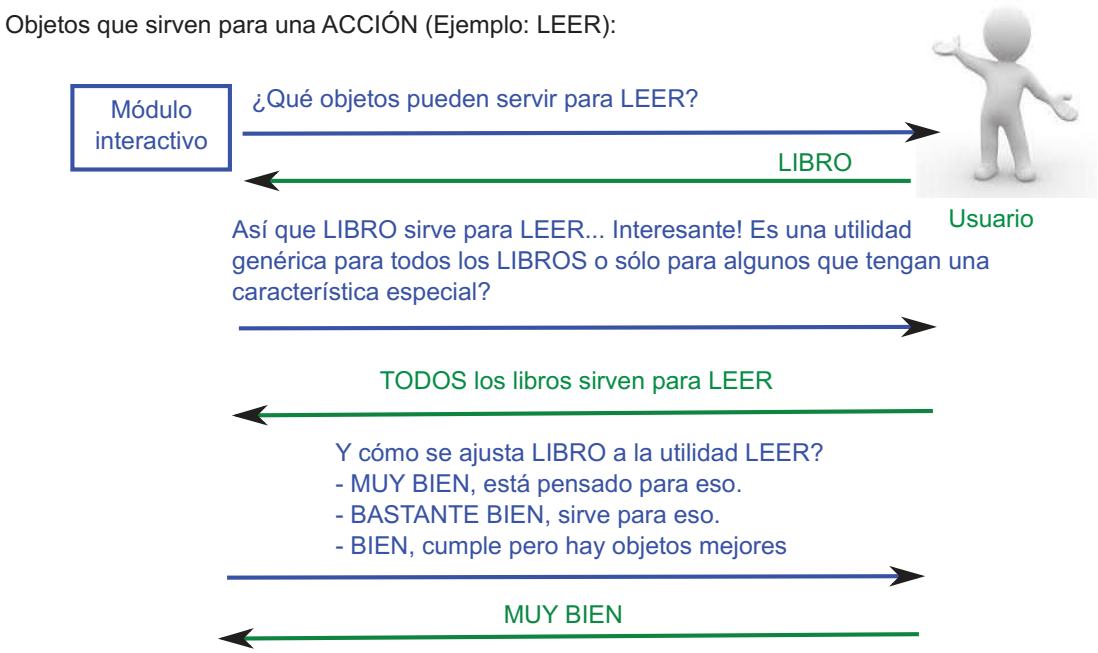


Figura 5.16: Ejemplo de secuencia de aprendizaje de objetos que sirven para una utilidad mediante diálogo.

El aprendizaje de acciones con significado añadido es más sencillo. Normalmente el sistema decide que debe hacer esta consulta cuando intenta acceder a un objetivo definido de esta manera pero se ha quedado sin opciones. Por ejemplo, si el usuario ha pedido ir a un sitio tranquilo y el sistema ya ha descartado los sitios que conoce, se iniciaría la secuencia de preguntar al usuario por acciones que sean tranquilas.

5.5.1. Implementación del módulo interactivo por teclado

La primera versión de este sistema de interacción fue implementada usando el teclado y mensajes de pantalla como interfaz con el usuario. El menú inicial simplemente consta de dos opciones: mandar un destino semántico y cerrar el programa.

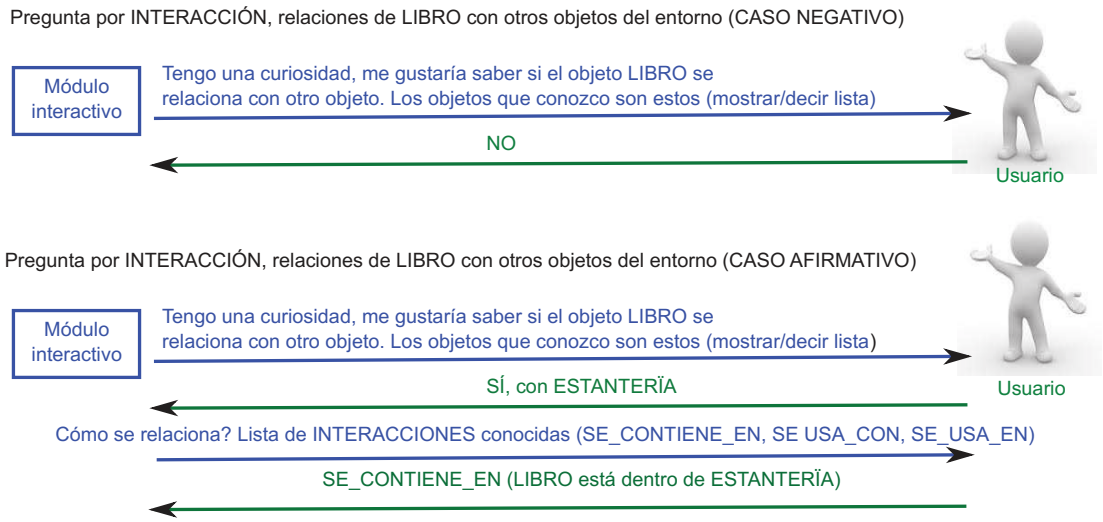


Figura 5.17: Ejemplo de secuencia de aprendizaje de objetos que interaccionan con otros objetos mediante diálogo.

No obstante, cuando aparece la necesidad por parte del sistema de preguntar al usuario, aparecen nuevas opciones por la pantalla en función de los datos que se desean obtener.

En estas opciones, el sistema vuelca información así como espera recibir los datos que necesita. Para facilitar la interacción, cuando el usuario debe responder con un objeto, se muestra la lista de objetos conceptuales conocidos. Si el objeto introducido es nuevo, se almacena como tal. Si no, simplemente se añaden las relaciones pertinentes. Lo mismo se hace con la lista de acciones o de interacciones. El usuario es guiado durante todo el proceso mediante mensajes por pantalla.

La filosofía de la implementación mantiene las funcionalidades de la interacción separadas del medio interactivo.

5.5.2. Implementación del módulo interactivo con diálogo de voz

La segunda versión del sistema de interacción fue implementada con un sistema de diálogo por voz. Esto llevó a la dirección de un Trabajo Fin de Máster cuyo autor

fue Enrique Moure y fue co-dirigido por Jonathan Crespo y Javier Fernández de Gorostiza [84].

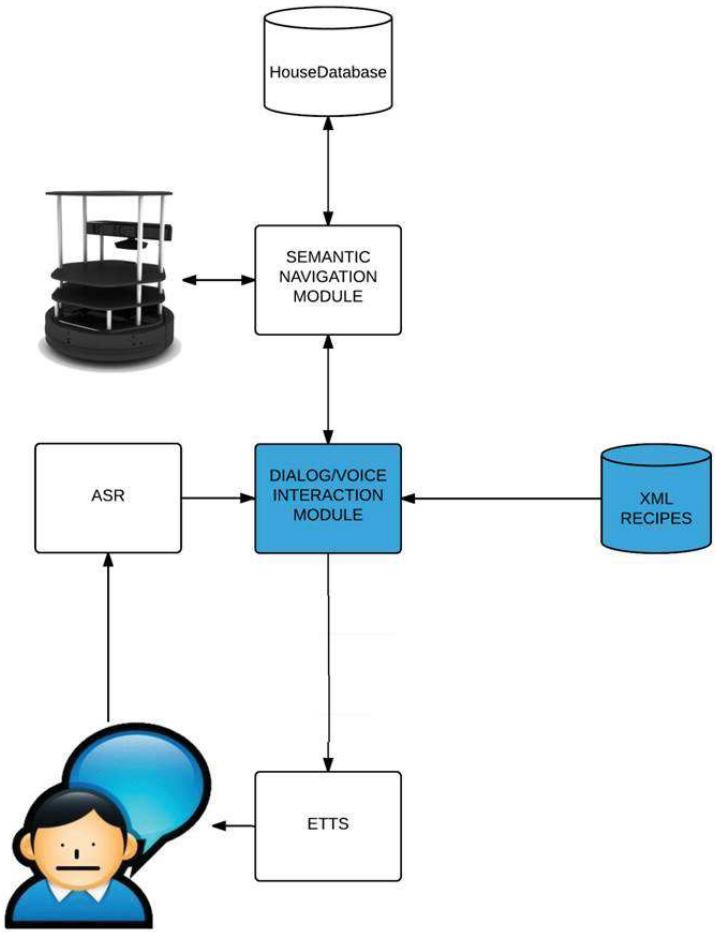


Figura 5.18: Diagrama del módulo de diálogo y sus relaciones con el resto del sistema y usuario.

El sistema de diálogo incluía definir una serie de recetas gramaticales para que el sistema interpretara la información y hablara de una forma más natural. En la figura 5.18 se muestra los elementos desarrollados en este TFM y la comunicación con el resto del sistema.

El funcionamiento fue diseñado para que siguiera el mismo procedimiento que la interacción por teclado, sin embargo los mensajes no se mostraban por pantalla ni se

introducían datos por teclado, sino que la información necesaria del usuario se recibía mediante diálogo del mismo modo que el sistema se comunicaba con voz.

5.5.3. Integración del módulo interactivo con el sistema de navegación semántica

Al diseñar el sistema de navegación semántica, ya se tuvo en cuenta la necesidad de incluir un subsistema de comunicación con el usuario que pudiera ser desarrollado utilizando diferentes vías de comunicación. Este subsistema se ha implementado siguiendo la arquitectura modular propuesta, en la que se considera de suma importancia evitar el acoplamiento. Esto se ha realizado con la intención de aportar las dos ventajas conseguidas:

- Facilidad para cambiar de módulo de interacción. Tan sencillo como ejecutar uno u otro.
- Integración sencilla, para poder crear nuevos sistemas de comunicación con el usuario.

En este caso, la facilidad de integración se ha llevado a cabo mediante el uso de ROS, sus topics y servicios. El planificador semántico es el que controla en cada momento si el robot debe intentar comunicarse con el usuario o no. Cuando decide que es necesario hacer una consulta al usuario, invoca una llamada a un servicio ROS. Los servicios ROS implementados que permiten aprendizaje mediante interacción son:

Accionesxsig Servicio que devuelve una lista de acciones que tienen un significado añadido concreto. Se envía el significado añadido y se espera la lista de acciones.

PreguntoPorObjetosConUtilidad Servicio que devuelve una lista de objetos que cumplen con una acción determinada. Cada objeto puede tener asociada una característica y todos tienen un grado que determina lo bien que se ajustan a esa acción o utilidad.

PreguntaPorInteraccion Servicio que devuelve una lista de objetos relacionados con otros objetos mediante algún tipo de interacción. Se envía un objeto y se espera recibir la lista de objetos relacionados con el objeto enviado.

El módulo interactivo ofrece otras funcionalidades no relacionadas con el aprendizaje que se comentan en la sección 3.5.2. En ambos casos la integración se consigue haciendo que los subsistemas atiendan a los mismos servicios y devuelvan el mismo tipo de información.

Este capítulo ha descrito la manera en la que se ha afrontado la adquisición de nuevo conocimiento por parte del sistema, tratando de dotar al robot de gran autonomía pero permitiendo cuando es necesario la comunicación directa con el usuario.

CAPÍTULO 6

Resultados experimentales

"La tarea cotidiana de la ciencia no consiste en cazar datos, sino en verificar hipótesis, es decir, en ver si resisten la prueba de la vida real o, cuando se trata de inventos, en ver si funcionan." — Peter Medawar

Es este capítulo se han recogido los experimentos más relevantes que se han llevado a cabo a lo largo de toda la tesis doctoral. Se incluyen pruebas para comprobar el correcto funcionamiento de cada módulo por separado, pruebas de integración, pruebas destinadas a determinar la eficiencia de distintas partes del sistema, pruebas de comparación entre distintas soluciones a un mismo problema y pruebas preliminares de algunas propuestas. La mayoría de experimentos han sido realizados sobre la plataforma robótica comercial conocida como Turtlebot (Figura 6.1).

6.1. Pruebas de funcionamiento del módulo de razonamiento

Estos experimentos comienzan poniendo a prueba el sistema de razonamiento implementado con la base de datos relacional. El objetivo en este caso es demostrar la

Figura 6.1: Turtlebot

funcionalidad del modelo y de su implementación. En estos experimentos, se obtienen destinos geométricos a partir de un destino semántico. Se parte de una supuesta situación donde el entorno ha sido parcialmente explorado. Esto se refleja en la información contenida en la tabla 6.1, con la información semántica de relaciones de la tabla 6.3 y con el conocimiento deducido almacenado en la tabla 6.2. El test se enfoca en el sistema de razonamiento, planificación y comunicación con el sistema de navegación de bajo nivel, por lo que toda la información den entorno reflejada en estas tres tablas ha sido introducida manualmente en la base de datos.

La base de datos ha sido creada con MySQL 5.5.43 para Ubuntu 12.04. El sistema ha sido creado dentro de un paquete de ROS (versión INDIGO) al cual se le ha añadido la dependencia *libmysqlclient-dev* para acceder a la base de datos desde el código en C++.

ObjetoFísico			
Nombre	**NombreConceptual**	**NombreEstancia**	**Posición**
Chair-1	SILLA	Estancia-2	4
Chair-2	SILLA	Estancia-2	5
Chair-3	SILLA	Estancia-3	6
Frigorífico-1	FRIGORÍFICO	Estancia-1	7
Sofá-1	SOFÁ	Estancia-2	8
Escritorio-1	ESCRITORIO	Estancia-3	9

Tabla 6.1: Contenido de la tabla ObjetoFísico

EstanciaFísicaConceptual	
NombreConceptual	**NombreFísico**
Estancia-1	COCINA
Estancia-2	SALÓN
Estancia-3	DESPACHO

Tabla 6.2: ConceptualPhysicalEstancia

EstanciaObjetoConceptual	
EstanciaName	**NombreObjeto**
COCINA	HORNILLO
COCINA	FRIGORÍFICO
COCINA	FREGADERO
COCINA	LAVAVAJILLAS
COCINA	RADIADOR
SALÓN	SOFA
SALÓN	MESA DE CAFÉ
SALÓN	SILLA
SALÓN	TELEVISOR
SALÓN	RADIADOR
SALÓN	ESTANTERÍA
COMEDOR	SILLA
COMEDOR	MESA DE COMEDOR
COMEDOR	RADIADOR
DESPACHO	RADIADOR
DESPACHO	SILLA
DESPACHO	ESTANTERÍA
DESPACHO	ESCRITORIO
DESPACHO	ORDENADOR
BAÑO	LAVABO
BAÑO	RETRETE
BAÑO	RADIADOR

Tabla 6.3: Contenido de la tabla EstanciaObjetoConceptual

6.1.1. Estancias Físicas

Esta prueba intenta analizar el comportamiento del planificador cuando se solicita llegar a una estancia conceptual que corresponde con una estancia física conocida.

Esto representa los casos en los que el usuario solicita ir a una estancia dando el nombre genérico de dicha estancia. Y el robot ya tiene el conocimiento de que al menos una de las estancias físicas del entorno se corresponde con ese tipo de estancia.

La prueba ha consistido en preguntar por la COCINA. La tabla 6.2 a la que debe acceder el robot, contiene la información de que la estancia 1 es una cocina. Por lo tanto estamos en el escenario deseado. El razonador identificó la petición correctamente y envió la consulta adecuada a la base de datos mediante el middleware implementado. Al realizar una consulta sobre la tabla *EstanciaFísicaConceptual* buscando la estancia física cuyo nombre conceptual es *cocina*, obtuvo en nombre *Estancia-1*.

Si el usuario responde que no es la estancia correcta, el sistema vuelve a repetir la búsqueda descartando el resultado anterior. Como no hay más cocinas en la tabla *EstanciaFísicaConceptual*, la petición se corresponde a otro escenario, puesto que la búsqueda ya no es por una estancia conceptual conocida. Esta situación delega el control al explorador.

6.1.2. Objetos conceptuales

Las pruebas descritas en esta sección evalúan cómo se comporta el sistema cuando se intenta encontrar un objeto conceptual cuando se ha observado un objeto real que corresponde con el objeto deseado. Este escenario representa la situación en la que el usuario solicita encontrar un objeto dando su nombre conceptual y el sistema conoce la ubicación de una o más instancias de ese objeto.

En el entorno inicial, existen tres instancias del objeto *silla*. Cuando se solicitó encontrar una silla, el sistema realizó la consulta adecuada contra la base de datos y obtuvo las tres sillas registradas en la tabla *ObjetoFísico*: *Silla-1*, *Silla-2* y *Silla-3*. Además, en la misma consulta se extrajo la información sobre la estancia física donde se encontraba el objeto y el identificador de la posición donde fue percibido. La primera silla se corresponde con el identificador 4 de la tabla de posiciones. Ese identificador se corresponde con una coordenada geométrica, que es la que fue transmitida al navegador de bajo nivel.

Después se continuó solicitando un objeto *silla*. Puesto que había tres sillas físicas en las tablas, se repitió el mismo proceso tres veces. Esto se refleja en la figura 6.2. Cuando se preguntó por una silla una vez más, el escenario cambió. Ya no se trataba de

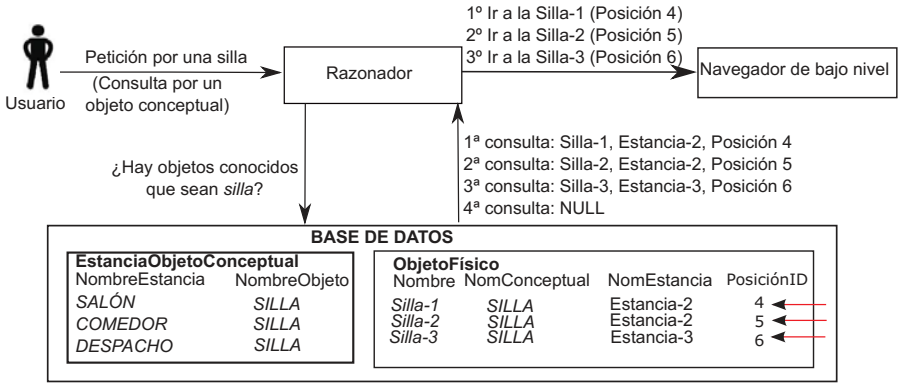


Figura 6.2: Secuencia de pruebas con el objeto SILLA

objetos conocidos, sino de objetos conceptuales no conocidos. El proceso es diferente porque el robot ya no se dirige hacia el lugar exacto donde sabe que hay un objeto físico, sino que tiene que deducir el lugar donde podría encontrarse un objeto de ese tipo. Esta información se obtiene de la tabla *EstanciaObjetoConceptual*. La tabla 6.3 representa lo que sucedió con una cuarta, quinta y sexta consulta sobre *silla*. El cambio de escenario hizo que el sistema consultase los lugares donde podía haber sillas y obtuvo como respuesta el salón, el comedor y el despacho. A continuación, el robot se dirige a esos lugares para explorar buscando sillas. En estas pruebas el comportamiento del explorador fue simulado de forma en la que siempre devolvía un resultado negativo, para comprobar cómo respondía el razonador.

6.1.3. Objetos no conocidos en estancias no conocidas

Esta prueba se lleva a cabo para asegurar el correcto funcionamiento en casos en los que se ha introducido un destino semántico en forma de objeto conceptual pero no ha sido observada ninguna instancia real de ese objeto. Además, en este escenario se supone que la posible ubicación de una instancia del objeto solicitado se encuentra en un tipo de estancia que tampoco ha sido identificada. Una manera ya vista de llegar a este escenario es continuando el experimento anterior solicitando una nueva silla cuando ya se han explorado los entornos conocidos.

El objeto solicitado es el *lavabo* y se recrea la situación descrita. La ubicación posible del lavabo es la estancia *baño* y este tipo de estancia no ha sido identificada,

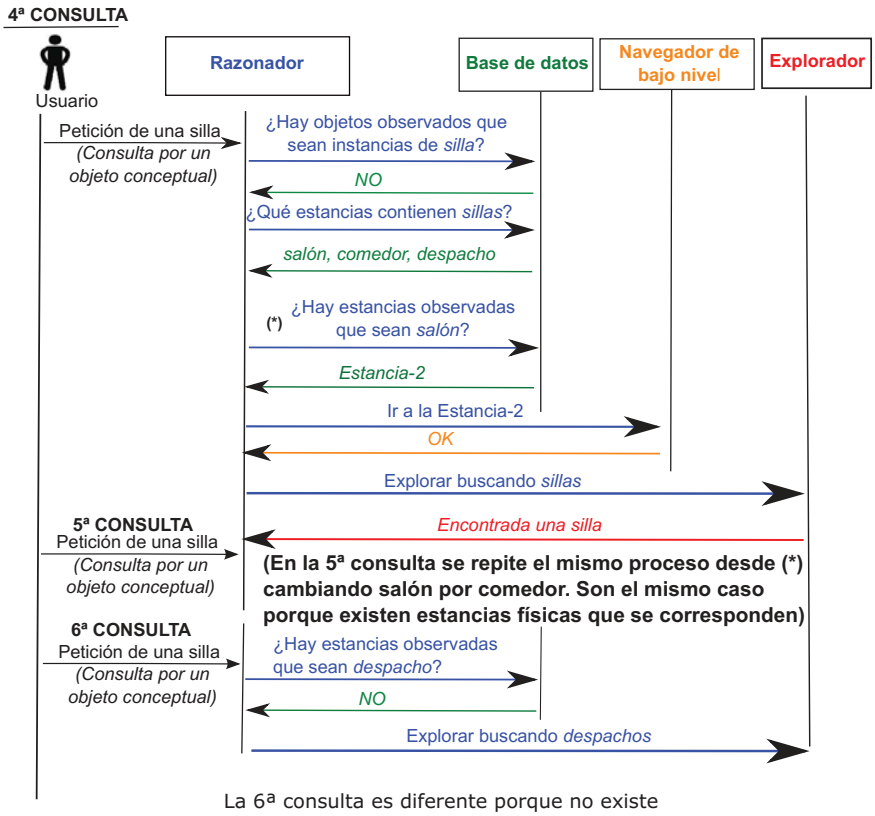


Figura 6.3: Secuencia de pruebas çon el objeto SILLA cuando no hay correspondencia con objetos físicos observados

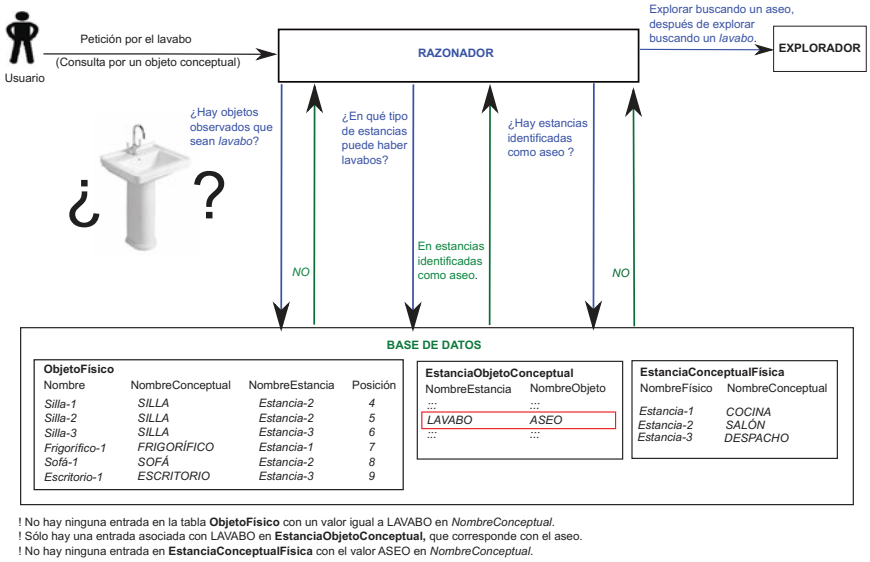


Figura 6.4: Secuencia desarrollada en el experimento de la búsqueda del LAVABO

tal como muestran las tablas del comienzo de esta sección. La figura 6.4 muestra el proceso ocurrido con esta petición en este entorno. El sistema decidió explorar el entorno para encontrar la estancia *baño* y de este modo, encontrar el lavabo.

6.1.4. Búsqueda de objetos por sus características.

En esta sección se comentan las pruebas realizadas en la recreación de un escenario en el que el usuario solicita encontrar un objeto con unas características determinadas. Se añade la información mostrada en la figura 6.5. Además, se añade la información de la tablas 6.4. Se le pide al sistema que encuentre algo *frío* para beber. En este caso, el sistema busca los objetos que tienen la utilidad *beber* y además la característica *frío*. Según la información almacenada, estos objetos son un *refresco* y *agua fría*.

En el caso del refresco, el sistema no cuenta con información en la tabla *EstanciaObjetoConceptual* para encontrar la posible ubicación del objeto. Por ello, se inicia el proceso de razonamiento que trata de relacionar unos objetos con otros. Mediante la información de la tabla *ObjetoCaracterísticaInteracción*, el sistema extrae la información de que un refresco frío sólo puede estar en el interior del frigorífico. En este momento, el experimento se ha convertido en encontrar un objeto desconocido en

ObjetoCaracterísticaInteracción			
NombreSujeto	**Característica**	**NombreInteracción**	**NombreObjeto**
AGUA	FRÍO	SE_CONTIENE_EN	FRIGORÍFICO
AGUA	CALIENTE	SE_CONTIENE_EN	FREGADERO
REFRESCO	FRÍO	SE_CONTIENE_EN	FRIGORÍFICO

Tabla 6.4: Información añadida a la tabla ObjetoCaracterísticaInteracción

una estancia conocida (*frigorífico* en la *cocina*). Este escenario ya había sido probado anteriormente. En esta ocasión el sistema también responde como se esperaba y emite la orden al navegador de bajo nivel de ir a la posición del frigorífico en la estancia cocina.

El último experimento de esta tanda, consiste en solicitar al sistema ir a un lugar relajante. El razonador extrae de la base de datos que *descansar* es relajante, así que busca objetos que sirven para descansar. Puesto que las sillas sirven para descansar, el experimento continua una vez más pidiendo una silla. El proceso a partir de aquí ha quedado descrito en ejemplos anteriores sobre búsquedas de objetos, cuando se solicita encontrar el objeto silla (secciones 6.1.2 y 6.1.3).

6.1.5. Búsqueda de objetos por proximidad semántica

InteracciónObjetoConceptual		
NombreSujeto	**NombreInteracción**	**NombreObjeto**
IMPRESORA	SE_USA_CON	ORDENADOR
REFRESCO	SE_CONTIENE_EN	ARMARIO_COCINA
CAFÉ	SE_HACE_CON	CAFETERA

Tabla 6.5: Información añadida a la tabla InteracciónObjetoConceptual

Estas pruebas están destinadas a comprobar la búsqueda de objetos mediante sus relaciones semánticas con otros objetos, de modo que buscando objetos relacionados se espera encontrar el objeto solicitado. Se vuelve a tener en cuenta la información añadida en las tablas de la figura 6.5 y además se incluye la información de la tabla 6.5.

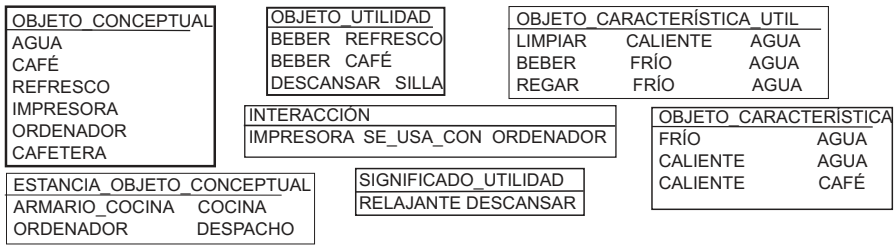


Figura 6.5: Conocimiento del robot representado en tablas.

Se solicita al sistema encontrar la impresora. Puesto que no hay información de la ubicación de este objeto, el razonador intenta buscar objetos relacionados semánticamente con la impresora. Según la información de la tabla 6.5, la impresora se utiliza con el ordenador, por lo que estos objetos están relacionados y sin ningún dato más, el sistema considera la opción de localizar el ordenador para encontrar la impresora. Esto vuelve a reducirse a un caso anterior, en este caso el de encontrar un objeto desconocido en una estancia conocida. El ordenador no había sido percibido anteriormente, pero según la tabla *EstanciaObjetoConceptual*, el ordenador se encuentra en el despacho. Y puesto que el despacho es una estancia conocida (la estancia-3), el razonador envía al planificador la posición asociada con la estancia-3 y el planificador envía la orden al navegador de bajo nivel.

6.1.6. Objetos desconocidos en estancias conocidas

En estas pruebas se recogen los resultados de solicitar objetos conceptuales que no han sido observados pero cuyas localizaciones se corresponden con estancias que sí han sido identificadas. Se ha comprobado que las inferencias se realizan correctamente y se envía al navegador de bajo nivel la posición adecuada. El objeto que ha sido elegido para estas pruebas es el *radiador*, el cual conceptualmente se encuentra en todos los tipos de estancia pero en la base de datos no consta que haya sido percibido ninguno. La respuesta del sistema fue explorar en todas las estancias conocidas, como se esperaba. Cuando se agotaron las estancias conocidas, el sistema ordenó explorar todo el entorno buscando un *radiador*.

6.1.7. Estancias desconocidas

Este último escenario se corresponde con la situación de solicitar ir a una estancia que aún no ha sido identificada, sin buscar ningún objeto concreto. En las pruebas se envió la petición de ir al *dormitorio*. Puesto que no había ningún dormitorio identificado, el escenario era el deseado. El razonador consultó la base de datos, descubrió que no había ningún dormitorio entre las estancias físicas identificadas y ordenó al explorador localizar este tipo de estancia, asumiendo este último el control del sistema. El dormitorio se diferencia del resto de estancias porque contiene el objeto *cama*, por lo que el explorador inició la búsqueda de este objeto por todo el entorno. Puesto que esta sección está centrada en el razonador, se interrumpe aquí el experimento y se da por válido el comportamiento del razonador.

6.1.8. Análisis de la eficiencia del sistema de razonamiento

En esta sección se recogen los experimentos destinados a analizar la eficiencia del razonador con el sistema implementado de acceso a la base de datos relacional. Los experimentos han sido enfocados a probar el razonador en situaciones concretas que pueden ocasionar un mayor coste computacional. Estos experimentos han sido realizados asumiendo que la información del entorno que tiene el robot es la misma que la que se presentó en las tablas 6.1, 6.2, 6.3 y la Figura 6.5.

Recursividad del sistema

Este sistema razonamiento está enfocado a la navegación de robots móviles en entornos de interiores. Partiendo de esta premisa, las situaciones que requieren una recursión profunda son escasas. Sin embargo, se ha considerado que estudiar cómo se desenvuelve el razonador afrontando operaciones recursivas es una interesante manera de aportar datos sobre la eficiencia. Estos experimentos mostraron que la eficiencia del sistema no se veía comprometida en situaciones que requerían múltiples llamadas recursivas. La recursión se manifiesta en llamadas a la función encargada de obtener el destino semántico cuyo pseudocódigo corresponde con el algoritmo 2 que se describió en la sección 4.3.2.

Destino solicitado: DESPACHO				
Iter	**Método**	**Entrada**	**Salida**	**Tiempo**
1	getEstanciaFísicaDeConceptual	DESPACHO	Estancia-2	6 ms
Tiempo para calculoDestino (puede incluir métodos no recursivos)				10 ms

Tabla 6.6: Resultados de tiempos de ejecución para llamadas que no necesitan recursión.

Destino solicitado: HOJA DE PAPEL				
Iter	**Método**	**Entrada**	**Salida**	**Tiempo**
1	buscarObjeto	PAPEL	IMPRESORA	8 ms
2	buscarObjeto	IMPRESORA	ORDENADOR	8 ms
3	buscarObjeto	ORDENADOR	DESPACHO	7 ms
1	getEstanciaFísicaDeConceptual	DESPACHO	Estancia-2	6 ms
Tiempo para calculoDestino (puede incluir métodos no recursivos)				36 ms

Tabla 6.7: Resultados de tiempos de ejecución para una llamada que provoca varias llamadas recursivas.

Esta función invoca a una serie de métodos que a su vez se vuelven a llamar a ellos mismos o a la función inicial, con lo que surge la recursión. Todas las llamadas a esta función son comentadas a continuación para discernir si la recursión podría ser un problema. Se han llevado a cabo experimentos de búsquedas básicas por una estancia conceptual, que sirve como referencia de llamada básica en contraposición a las llamadas recursivas. Los datos de tiempo relevantes que pueden afectar a la recursión son mostrados en la tabla 6.6. En este primer experimento se ha ordenado al sistema que encuentre el despacho y esta llamada no genera ninguna recursión, con lo que se emplean estos datos como referencia para el siguiente experimento. El segundo experimento es una consulta que conlleva tres niveles de recursión. Consiste en solicitar al sistema que encuentre una hoja de papel. Para esto se ha añadido la información de una nueva interacción que relaciona los objetos *papel* con *impresora* mediante el tipo de relación *se_usa_con*. Los resultados se muestran en la Tabla 6.7.

También se ha realizado el mismo análisis con los experimentos de la sección 6.3, puesto que surgieron llamadas recursivas de manera natural. Los resultados se muestran en la tabla 6.8. Estas comparaciones muestran que el sistema permite y maneja la recursión de una manera competente puesto que los tiempos de ejecución se

Destino solicitado: UN LUGAR DIVERTIDO				
Iter	**Método**	**Entrada**	**Salida**	**Tiempo**
1	buscarPorSignificado	DIVERTIDO	LIBRO/T.V.	9 ms
1	buscarObjeto	LIBRO	ZONA_LECTURA	3 ms
1	getEstanciaFísicaDeConceptual	ZONA_LECTURA	Posición	8 ms
Tiempo para calculoDestino (puede incluir métodos no recursivos)				34 ms

Tabla 6.8: Resultados de tiempos de ejecución para una llamada que provoca varias llamadas recursivas.

incrementan normalmente. El tiempo de la llamada simple de referencia al algoritmo de cálculo de destino fue de 10 ms, y la llamada interna de 6 ms. En el resto de experimentos, las llamadas internas oscilaron entre 6 y 9 ms. Esto indica que el coste de las llamadas internas no crece con la recursión. Del mismo modo, un coste de 36 ms para una llamada recursiva de 3 niveles (más una llamada no recursiva) a la función de cálculo de destino cabe dentro de lo esperado, pues debe albergar el coste de 4 llamadas a funciones y el coste de una llamada básica fue de 10 ms. Por todo esto, se considera que el sistema resuelve adecuadamente el problema de la recursión.

Tiempos de procesamiento en grandes entornos

Este apartado se centra en comprobar cómo responde el sistema de razonamiento cuando la base de datos contiene más información. Los resultados no han mostrado ningún problema cuando la base de datos estaba más cargada de contenido. Esto significa que el sistema mantiene sus prestaciones de eficiencia cuando se le presentan entornos más grandes y con un mayor número de entidades.

Se han tenido en cuenta los datos de tiempos de procesamiento relevantes de la sección 6.3 como valores de referencia. Posteriormente se introdujeron muchos más datos en varias tablas con un software destinado a sobrecargar la base de datos y se repitió la misma secuencia del experimento. Los resultados de la comparación se muestran en la tabla 6.9. En esa misma tabla se indica la función involucrada, el tamaño de la tabla implicada en el primer experimento, el tiempo de procesamiento del primer experimento, el tamaño de la tabla en el segundo experimento y el tiempo de procesamiento del segundo experimento. Se han elegido las funciones más críticas y repetidas del experimento para este estudio. La función 1 se llama para deducir los

ID	Función con consulta	Tamaño	Tiempo	Tamaño	Tiempo
1	búsquedaPorSignificado	10	9 ms	55	11 ms
2	buscarObjeto	35	3 ms	115	3 ms
3	getEstanciaFísicaDeConceptual	3	8 ms	99	8 ms
	Tiempo total		20 ms		22 ms

Tabla 6.9: Resultados de los tiempos de procesamiento de una llamada que realiza varias consultas.

objetos que están relacionados con utilidades que son *divertidas*. La respuesta es *libro* (porque el sistema tiene la información de que leer es divertido) y *T.V.* (porque el sistema también contempla que ver la T.V. es divertido). El tiempo de procesamiento cuando la base de datos contiene 10 entradas en la tabla implicada es muy similar al tiempo de procesamiento cuando hay 55 entradas. Los experimentos arrojaron un resultado de 9 ms en el primer caso y de 55 ms en el segundo. Esto significa que para una carga de entradas 5.5 veces mayor, el tiempo sólo aumenta 1.2 veces en esa función. La función 2 es llamada para encontrar la localización del objeto *libro* y concluye que se debería encontrar en la *zona de lectura*. Aquí el tiempo medido fue exactamente igual para una carga de 35 entradas y para 115 entradas. Por último, el razonador consulta mediante la función 3 cuál es el área física del entorno que se corresponde con la sala de lectura, con lo que obtiene la posición de ese área. En este caso, también se registró el mismo tiempo de procesamiento cuando la carga de la tabla implicada era de 3 entradas y cuando era de 99 entradas.

Esto quiere decir que para una consulta de destino que ha movido en total 48 elementos, el tiempo ha sido de 20 ms. Y con la misma consulta pero gestionando 269 elementos (5.6 veces más elementos), el tiempo ha sido de 22 ms (1.1 veces más tiempo).

Con esto se concluye que el razonamiento accediendo a una base de datos no sufre alteraciones de rendimiento apreciables con grandes entornos.

6.1.9. Comparación con razonador basado en KnowRob

El sistema de razonamiento ha sido implementado con dos tecnologías diferentes. Una de ellas está basada en el acceso a una base de datos relacional que ha sido

diseñada expresamente para almacenar los conceptos necesarios que describen un entorno, sus entidades y sus relaciones. La otra está basada en el sistema KnowRob que han desarrollado en el Instituto de Inteligencia Artificial en la Universidad de Bremen, Alemania.

La integración de ambos sistemas de razonamiento es sencilla debido a que ambos razonadores heredan de una superclase que con sus métodos ya deja definida la manera en la que el resto del sistema de navegación interacciona con el razonador. Las pruebas que se han llevado a cabo tienen el objetivo de comprobar por un lado que la información extraída de ambos sistemas de razonamiento es la misma y por otro lado, se intenta comparar en eficiencia.

Descripción de las pruebas

Para probar el sistema de razonamiento basado en KnowRob y poder compararlo con el desarrollado en esta tesis, se ha creado un programa que realiza una batería de pruebas sobre ambos razonadores. Las pruebas consisten en probar cada uno de los métodos que ofrece el razonador, siendo cada método una manera de consultar información a través del mismo. La información pedida es introducida por el usuario que está realizando las pruebas. Aunque en el sistema real son el planificador o el explorador los que interactúan con el razonador, en este programa de pruebas es necesario hacer las consultas manualmente para probar cada método. Además, este programa permite activar el razonamiento con KnowRob y con el sistema basado en la base de datos relacional indistintamente, así como indicar el número de veces que debe ser repetida cada operación. Esto se debe a que se ha añadido un parámetro para que las consultas se realicen un número determinado de veces y de este modo el tiempo de ejecución es la media de esas veces. Esto elimina la posibilidad de que por azar una consulta saliera con un tiempo mucho mayor o menor de lo que realmente debería salir. En estas pruebas se estableció un parámetro de 10 ejecuciones.

Ademas hay que tener en cuenta que:

- La ontología para el razonadorKR fue creada con Protege versión 5.1.
- El servidor de consultas en Prolog usa la versión de swi-prolog 6.6.

La información contenida en ambos razonadores es la mostrada en la figura 6.6

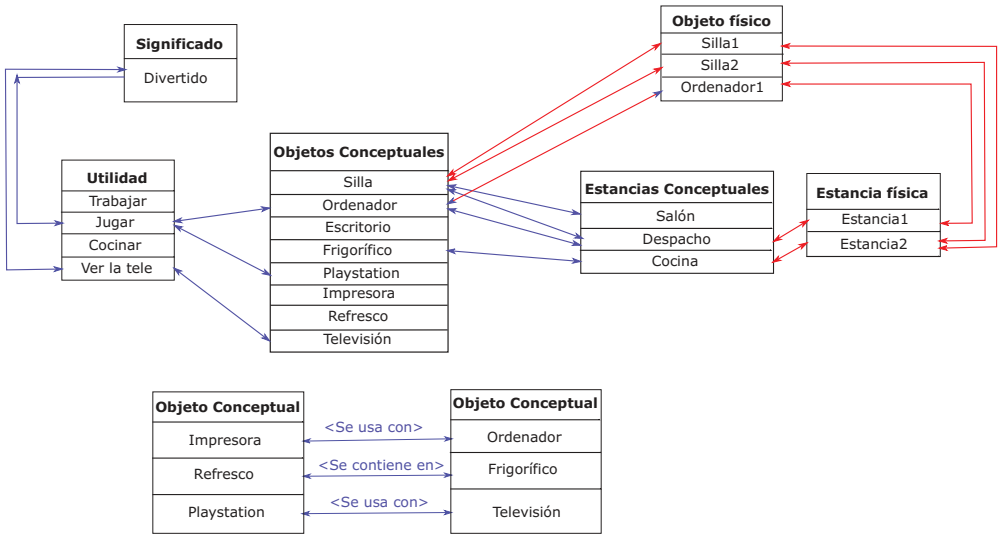


Figura 6.6: Conocimiento sobre el que se realiza el experimento de comparación entre razonadores. Las líneas rojas corresponden a datos sobre el mundo real y las moradas a relaciones conceptuales.

Resultados

La primera cuestión que pretende responder esta comparativa es si ambos sistemas obtienen las mismas conclusiones. En caso afirmativo, sería interesante comparar los tiempos de ejecución. El programa de pruebas permite ir ejecutando cada método una serie de veces con una entrada que se introduce por teclado. El programa envía la entrada a cada método de cada razonador seleccionado y guarda en un archivo los tiempos de ejecución.

La tabla 6.10 contiene la información relativa a las entradas que se indicaron a cada método, así como las salidas que estos devolvieron. Se observa que ambos razonadores concluyen lo mismo, con la única particularidad de que cuando una consulta devuelve varias entidades como respuesta, el orden de estas entidades no tiene por qué coincidir. Esto es normal porque con la ontología diseñada para KnowRob no se establece ningún criterio para la ordenación.

Los tiempos medios de ejecución son mostrados en la tabla 6.11. El número de veces que se repitió cada consulta fue de 10. Como se puede apreciar a simple vista, el tiempo de ejecución con el sistema que accede a la base de datos relacional fue

		Salida del método	
Método	**Entrada**	**SalidaBD**	**SalidaKR**
Etiquetado semántico por objeto	Ordenador	Despacho	Despacho
Estancia conceptual de estancia física	Estancia2	Cocina	Cocina
Estancias conceptuales que contienen un objeto	Silla	Salón Despacho	Salón Despacho
Objetos relacionados semánticamente	Refresco Impresora	Frigorífico Ordenador	Frigorífico Ordenador
Objetos con un uso concreto	Trabajar	Ordenador	Ordenador
Objetos cuyo uso tiene asociado un significado	Divertido	Playstation Televisión Ordenador	Ordenador Playstation Televisión
Localización probable de un objeto	Refresco	Cocina (F)	Cocina (F)
Estancia física de estancia conceptual	Despacho	Estancia1	Estancia1
Objetos contenidos en estancia física	Estancia1	Silla Ordenador	Silla Ordenador
Objetos físicos que son objeto conceptual	Silla	Silla1 Silla2	Silla1 Silla2
Nombre conceptual de objeto físico	Silla1	Silla	Silla
Obtener todos los objetos conceptuales	—	Lista (*)	Lista (*)
Obtener todas las acciones/utilidades	—	Lista (**)	Lista (**)

Tabla 6.10: Resultado de la comparación de las salidas entre razonadores con distintas tecnologías. (F): Frigorífico. (*) Lista completa de objetos conceptuales, la ordenación varía según el razonador, en total cada razonador devuelve 8 objetos. (**) Lista de acciones, la ordenación varía, cada razonador devuelve 4 acciones

considerablemente menor que el tiempo de ejecución del sistema basado en KnowRob. Con los datos tomados y mostrados en la tabla, se concluye que de media de todas las ejecuciones en el experimento, las consultas del sistema diseñado y desarrollado en esta tesis tardaban un 8.9 % del tiempo que consumía una consulta con el sistema basado en KnowRob, lo que implica que el sistema basado en la base de datos es **11.13 veces más rápido**. Esto se conlleva que se liberen recursos de la máquina y se aumente el rendimiento del programa. Además es importante señalar que los experimentos han sido realizados con muy poca información, pero los datos sugieren que el sistema de KnowRob es más susceptible a ver incrementados sus tiempos cuando las consultas son más complejas o devuelven más resultados, como es el caso de los métodos de

	Tiempos promedios de ejecución	
Método	**TiempoBD**	**TiempoKR**
Etiquetado semántico por objeto	4 ms	17 ms
Estancia conceptual de estancia física	3 ms	20 ms
Estancias conceptuales que contienen un objeto	2 ms	29 ms
Objetos relacionados semánticamente	3 ms	50 ms
Objetos con un uso concreto	3 ms	16 ms
Objetos cuyo uso tiene asociado un significado	2 ms	63 ms
Localización probable de un objeto	4 ms	49 ms
Estancia física de estancia conceptual	3 ms	19 ms
Objetos contenidos en estancia física	3 ms	21 ms
Objetos físicos que son objeto conceptual	2 ms	20 ms
Nombre conceptual de objeto físico	2 ms	27 ms
Obtener todos los objetos conceptuales	4 ms	49 ms
Obtener todas las acciones/utilidades	2 ms	32 ms

Tabla 6.11: Resultado de la comparación de los tiempos entre razonadores con distintas tecnologías

Objetos cuyo uso tiene asociado un significado donde el tiempo medio sube a 63 ms, *Objetos relacionados semánticamente* donde el tiempo llega a 50 ms o *Localización probable de un objeto* donde se alcanzan 49 ms. Sin embargo, con el sistema basado en la base de datos, apenas se nota que esas consultas son más pesadas. La eficiencia de este sistema también se puso a prueba de forma independiente en la sección 6.1.8.

6.2. Integración de sistemas

El navegador desarrollado en esta tesis abarca múltiples módulos que han sido integrados cuidadosamente. Algunos, como el subsistema de interacción con el usuario, se han integrado con la metodología de utilizar ROS como punto común y ofrecer los mismos topics y servicios. Pero otros sistemas requieren de una sección dedicada a las pruebas de su integración por depender de métodos creados en esta tesis, aparte por supuesto del denominador común que representa ROS.

6.2.1. Integración con navegador de bajo nivel

El sistema de navegación semántico desarrollado en esta tesis ha sido diseñado con la idea de poder ser utilizado por cualquier navegador topológico o geométrico implementado en ROS. En esta sección se describen los experimentos realizados con el objetivo de comprobar esta ambivalencia.

El módulo navegador de bajo nivel está preparado para ser configurado introduciendo los topics que se han supuesto indispensables para el funcionamiento de un navegado topológico y geométrico. Además se debe indicar el tipo de navegador que es. Por lo tanto los parámetros a configurar son:

- Topic de petición de destino. Cualquier navegador necesita saber a dónde tiene que desplazarse. En un navegador implementado en ROS, esta información es transmitida por un topic. Ese topic es uno de los parámetros que necesita el módulo navegador de bajo nivel.

- Topic de destino alcanzado. Del mismo modo, los navegadores transmiten la información de que han alcanzado el destino solicitado. Si están implementados en ROS, esta información se publica en un topic.

- Topic de posición actual. Uno de los requisitos de la navegación es que el robot sepa dónde está. Por lo tanto esta información no puede faltar en ningún navegador y se publica en un topic siempre que esté implementado en ROS.

- Tipo de navegador. El módulo navegador de bajo nivel necesita ser configurado con el tipo de navegación que se integra, necesita distinguir si es un navegador geométrico o topológico.

Las pruebas han sido realizadas en un entorno doméstico en la planta inferior de un piso de estudiantes. Los tipos de estancia contemplados en esta prueba han sido la cocina, el salón y un pasillo. Para facilitar las pruebas y enfocarlas únicamente a la navegación de bajo nivel, los objetos del entorno se introdujeron manualmente con su posición.

La orden a ambos navegadores ha sido la misma y consistía en ir a la cocina desde el salón. Ambos navegadores llevaron al robot a la cocina (Figura 6.7).

Figura 6.7: Posición de salida del robot en los experimentos de la integración con el navegador de bajo nivel.

Ejecución con navegador geométrico

En esta prueba, se ha elegido el navegador geométrico basado en el algoritmo AMCL http://wiki.ros.org/amcl/ que viene por defecto en los tutoriales de generación de mapas geométricos con el turtlebot.

Los parámetros han sido:

- Topic de petición de destino: move_base_simple/goal
- Topic de destino alcanzado: move_base/result
- Topic de posición actual: odom
- Tipo de navegador: Geométrico.

El mapa generado por este navegador se muestra en la figura 6.8, se ha indicado sobre dicha figura los tipos de estancia que forman el entorno. El desplazamiento con este navegador fue suave y eficiente.

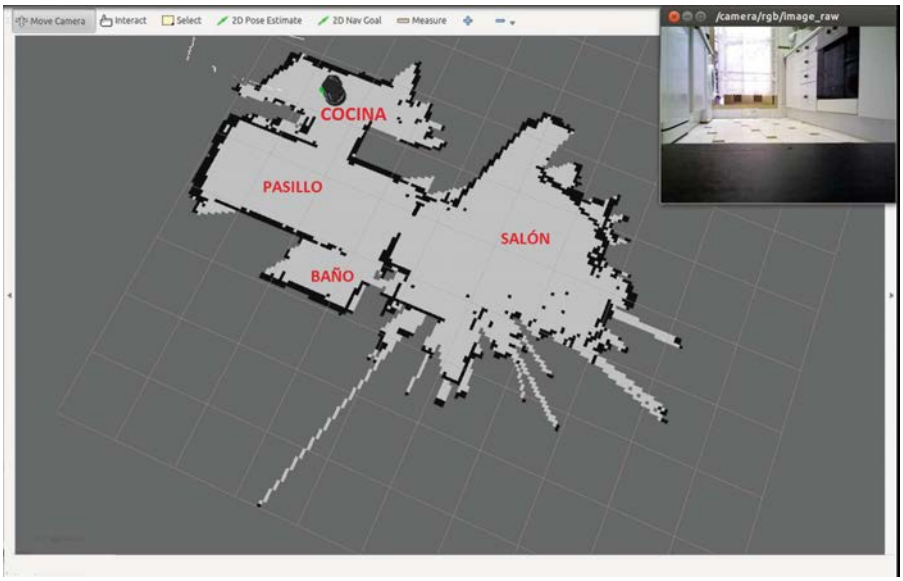


Figura 6.8: Imagen de lo que mostraba la pantalla de la estación de trabajo durante los experimentos de integración con el navegador de bajo nivel.

Ejecución con navegador topológico

Para probar la integración con un navegador topológico, se eligió el navegador desarrollado en el Trabajo Fin de Máster de Clara Gómez [14], que fue co-dirigido por el autor de esta tesis doctoral.

Los parámetros han sido:

- Topic de petición de destino: nodo_destino
- Topic de destino alcanzado: topoNavSuccess
- Topic de posición actual: topo_nav/position
- Tipo de navegador: Topológico.

En el estado de desarrollo del navegador topológico, el mapa fue introducido a priori, lo que no quita generalidad. El mapa introducido se describe con las instrucciones a continuación:

0{0 1}GP|odometry<0 -1.53>

Figura 6.9: Representación del mapa topológico cargado en el robot.

1{0 2}GP|odometry<1.61 0>
2{2 3}GP|odometry<3.45 -0.1>
3{3 4}GP|odometry<3.45 -1.60>
4{4 5}GP|odometry<5.0 -1.70>

Este mapa topológico contiene la información esquemática para que el robot se desplace de un nodo a otro. Las instrucciones que lo componen son íntegramente directrices de movimiento de la habilidad de ir a un punto (GP). En la figura 6.9 se muestra la representación de este mapa con las coordenadas de los nodos que contempla. Se indica con etiquetas los lugares a los que corresponden los nodos, habiendo uno de ellos en el salón, otro en la puerta que comunica el salón con el pasillo, otro en medio del pasillo a la altura de la puerta que comunica con la cocina y otros dos nodos dentro de la cocina, siendo el que tiene una mayor profundidad en el eje X el nodo asociado a la cocina.

El desplazamiento con este navegador fue efectivo, llevando al robot a la cocina, aunque se observó la necesidad de depurar un poco más el movimiento.

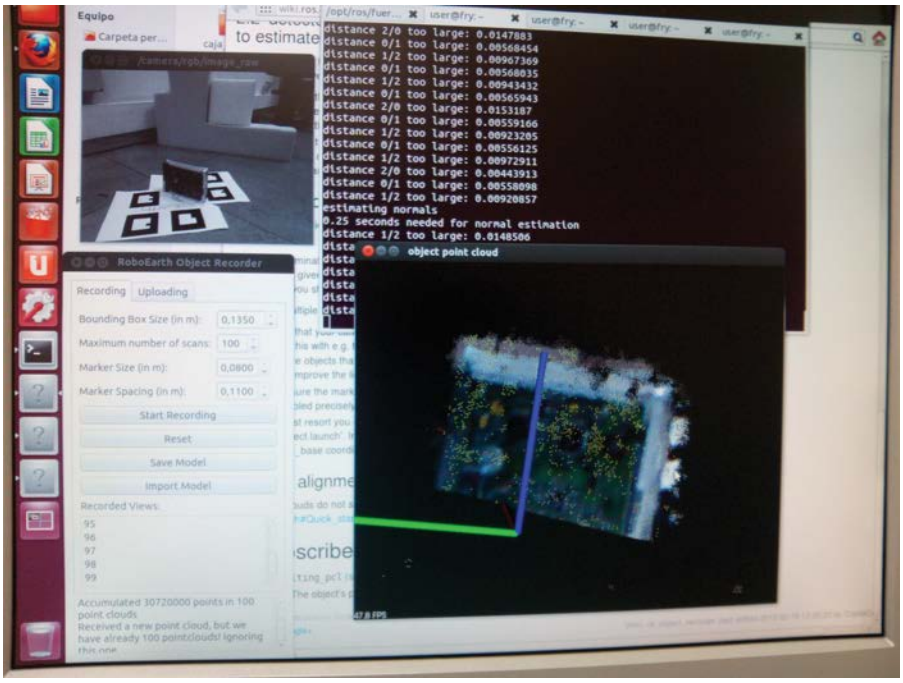


Figura 6.10: Modelo tomado para reconocer un disco duro.

6.2.2. Integración con sistemas de percepción visual

El navegador semántico requiere de la capacidad de percibir e identificar objetos de su entorno, por lo que dotar al sistema de esta capacidad ha sido una de las líneas de investigación prioritarias. Se han llegado a utilizar hasta cuatro sistemas diferentes de reconocimiento de objetos. Con la tecnología de RoboEarth, con detección de objetos etiquetados, con una técnica de combinación de detectores de contornos con descriptores y por último, con un clasificador de objetos usando SVM. En este apartado se comentan los experimentos de integración de estos sistemas de detección de objetos en el navegador.

Integración con RoboEarth

La primera integración con un sistema de percepción visual fue con RoboEarth. Se eligió dicho sistema porque ofrecía una funcionalidad adecuada a las exigencias del navegador y además porque estaba construido en ROS y eso facilitaba la integración. Las pruebas realizadas no cubrieron las expectativas de funcionamiento y el problema

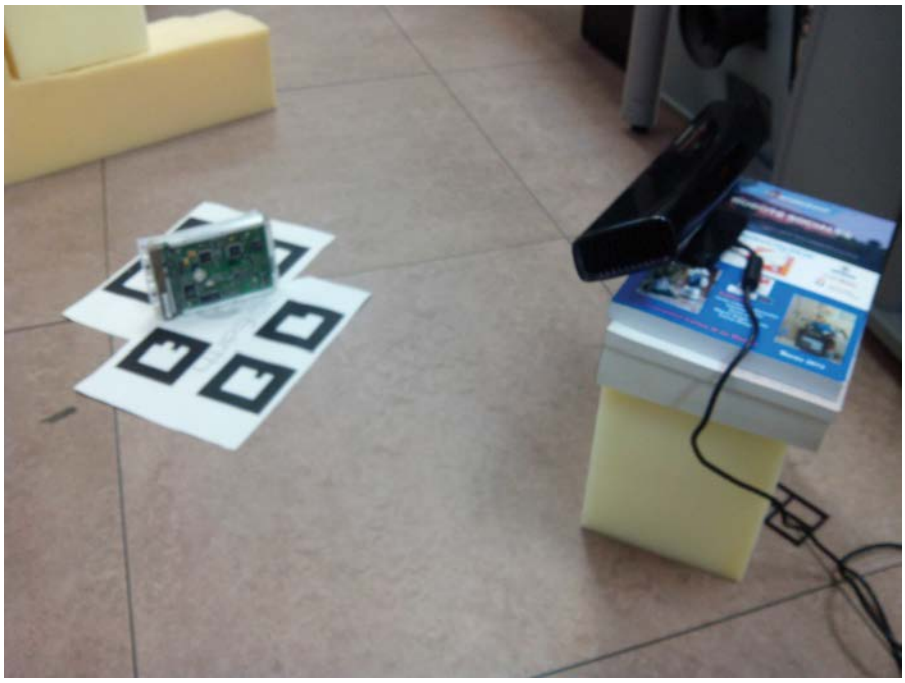


Figura 6.11: Modelando el disco duro.

de la detección de objetos tuvo que resolverse con otras técnicas. Los experimentos con RoboEarth comienzan con el modelado, para el cual se tiene que colocar el objeto sobre unas marcas donde al menos tres de ellas deben estar siempre visibles. Luego se le va dando vueltas al objeto y tomando valores de nubes de puntos que percibe la cámara. Cuando el modelo tiene suficientes puntos el sistema permite guardarlo. En los expermientos se emplearon tres modelados distintos para el mismo objeto, con densidades de puntos diferentes. RoboEarth emplea el paquete re_object_recorder para crear y almacenar modelos de objetos. Este paquete permite ir añadiendo al modelo nubes de puntos percibidas con la kinect, hasta un máximo de 100 nubes de puntos por objeto.

En la figura 6.10 se muestra la imagen del objeto generada al acumular 31 nubes de puntos. La interfaz no permite guardar un modelo si no hay suficiente densidad de puntos. Para comprobar la efectividad de este sistema de detección de objetos, se crearon tres modelos sobre el mismo objeto, colocándolo como muestra la figura 6.11, variando la densidad de puntos por si esto fuera determinante para la eficacia de la

detección. En uno de ellos se llegó al máximo que permitía la interfaz, **30.720.000** puntos acumulados en 100 nubes de puntos.

Durante las pruebas se hicieron las siguientes observaciones:

- El modelado es tedioso, presentándose dificultades a la hora de generar correctamente las nubes de puntos. En ocasiones el sistema tenía problemas para separar el suelo del objeto modelado, pese a que estaba bien colocado sobre las marcas. Las muestras erróneas ralentizaban el proceso de modelado.

- La detección es inestable. El sistema demostró tener baja fiabilidad. Se realizó la prueba de apuntar con la cámara directamente al objeto, tratando de simplificar al máximo el problema para luego ir creciendo en complejidad. Sin embargo, el sistema de detección no funcionó bien ni en las pruebas simplificadas. La figura 6.12 está tomada de una serie de vídeos que registraron las pruebas, recoge un momento en el que el sistema identifica el objeto y aparece marcado en rojo en la pantalla. Pero esto sucedía muy ocasionalmente, dejando pasar muchas mediciones sin detectar el objeto. En las pruebas se mantuvo control de las mediciones. Ante las dificultades del sistema para detectar el objeto, se probó a mover lentamente la kinect hasta que el sistema detectara el objeto y entonces mantener la cámara en la misma posición. Y repetir esto varias veces. Incluso en este experimento tan simplificado, en el mejor caso se obtuvieron 5 detecciones y 46 mediciones que no percibieron el objeto. En otro experimento, haciendo un lento barrido de la cámara sobre el objeto, se tomaron 35 mediciones y sólo una detectó el objeto.

Ante estos resultados, se descartó el empleo del este sistema. El sistema de navegación requiere que el robot tenga la capacidad de detectar objetos según se va desplazando y se consideró muy difícil de conseguir con esta tecnología.

Integración con ARToolkit

La integración con ARToolkit se basa en el empleo de ROS como medio común. La herramienta utilizada viene de un paquete de ROS que ofrece un *launcher* que arranca un nodo que publica la información del conjunto de etiquetas detectadas en

Figura 6.12: Pruebas con RoboEarth.

cada momento. Esa información incluye la posición, la orientación y el identificador de la etiqueta.

Esa información abre muchas posibilidades y se hicieron bastantes pruebas orientadas a obtener mediante funciones de transformación del marco, las posiciones de los objetos que representaban. La figura 6.13 es una captura de pantalla que se tomó en una de estas pruebas y tenía como objetivo conseguir las coordenadas en el suelo de una etiqueta que se correspondía a una silla. Además se probó con la idea de obtener también posición de *FRENTE_A_OBJETO*, aunque posteriormente se desestimó trabajar con ese dato puesto que lo que se buscaba era saber dónde estaban los objetos realmente. Si se contemplaba la información de la posición frente a un objeto, esta posición no sería siempre útil puesto que se situaría en frente de la etiqueta, no en frente del objeto. Y cuando el objeto no es estático podría dar problemas. De este modo se cumple lo que se necesita de un sistema de detección de objetos: identificar el objeto y ubicarlo espacialmente.

Para terminar de integrar este sistema como un detector de objetos, es necesario asociar el identificador de la etiqueta con un objeto conceptual. Había muchas maneras de hacer esto, al final se ha optado por almacenar esa información en la base de datos.

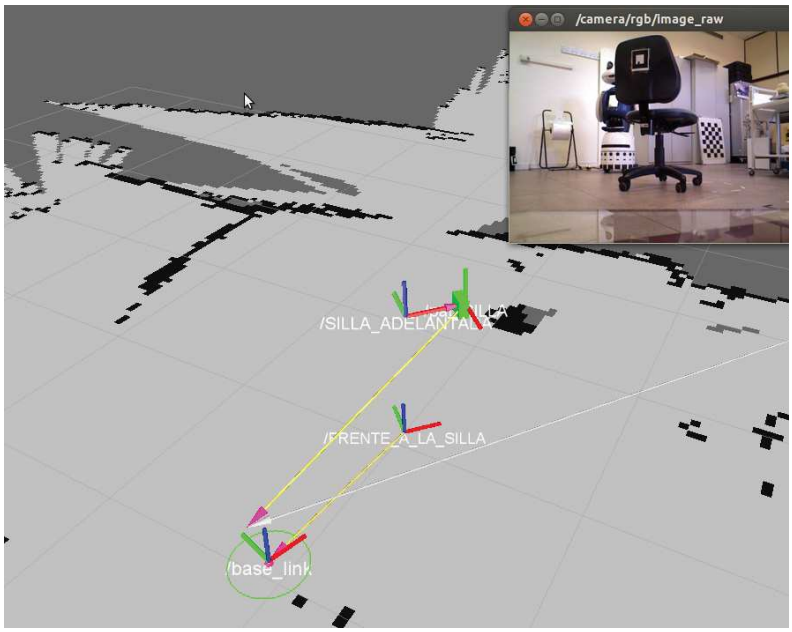


Figura 6.13: Detección de silla etiquetada con ARToolkit.

Primer sistema de detección de objetos. Unión de contornos y descriptores.

La integración con el primer sistema de detección de objetos diseñado bajo la perspectiva de ser utilizado en la navegación semántica no resultó ser completamente satisfactoria por mucho tiempo. Esto es porque las librerías utilizadas en ese momento para el sistema de detección de objetos estaban soportadas por ROS *Electrik*, mientras que la versión de ROS que estaba en el ordenador de la base se portó a *Groovy*.

Un equipo puede tener varias versiones de ROS instaladas, pero sólo un *core* corriendo a la vez y por lo tanto sólo un *máster*. Teniendo en cuenta que el diseño aconsejaba que el sistema de reconocimiento de objetos estuviese corriendo en el mismo ordenador que iba en la base, la integración era un problema.

El procesamiento de imágenes se realizaba en una máquina distinta a la que capturaba las imágenes. Esto hacía que el retardo fuese considerable y producía que los objetos fuesen detectados tarde. Por ello el diseño colocaba al reconocedor de objetos en el mismo ordenador de la base. Pero la incompatibilidad de versiones de ROS impedía que esto fuera viable.

Estos problemas animaron a continuar investigando en nuevas técnicas de detección de objetos.

Segundo sistema de detección de objetos. Clasificador con SVM.

El segundo sistema de detección de objetos presenta una compatibilidad plena con las versiones de ROS instaladas en el ordenador de la base. Tras *Groovy*, se migró a *Indigo* y ya no se volvió a cambiar de versión. El sistema actual de detección de objetos presentó otros problemas de integración. Técnicamente era posible que esta vez sí estuviera corriendo el detector en la misma máquina que movía la base y tenía conexión a la cámara (evitando el retardo en el procesamiento de imágenes). No había ninguna incompatibilidad y en los experimentos los sistemas respondían sin problemas. Pero en la práctica el sistema no funcionaba adecuadamente, se apreciaba un elevado uso de los recursos de la máquina y todos los procesos experimentaban malfuncionamiento por retardos. Esto hizo considerar que tal vez el equipo de la base no fuese lo suficientemente potente o que el algoritmo de detección de objetos no estuviese suficientemente optimizado.

La integración se ha resuelto añadiendo un ordenador de dedicación exclusiva a la detección de objetos, este ordenador se embarca directamente en el robot y se conecta a una cámara. El procesamiento de la imagen se realiza en este equipo y lo que se transmite como topic es la información procesada consistente en un identificador del objeto detectado y su posición. El tamaño del mensaje de este topic es varios órdenes de magnitud menor que el de una imagen, por lo que no se aprecia retardo significativo y el sistema detecta los objetos donde se supone que están. Este sistema es el utilizado en los experimentos de exploración de la sección 6.4.

6.3. Pruebas de aprendizaje

En esta sección se recopilan las pruebas realizadas sobre la capacidad del sistema de adquirir nueva información.

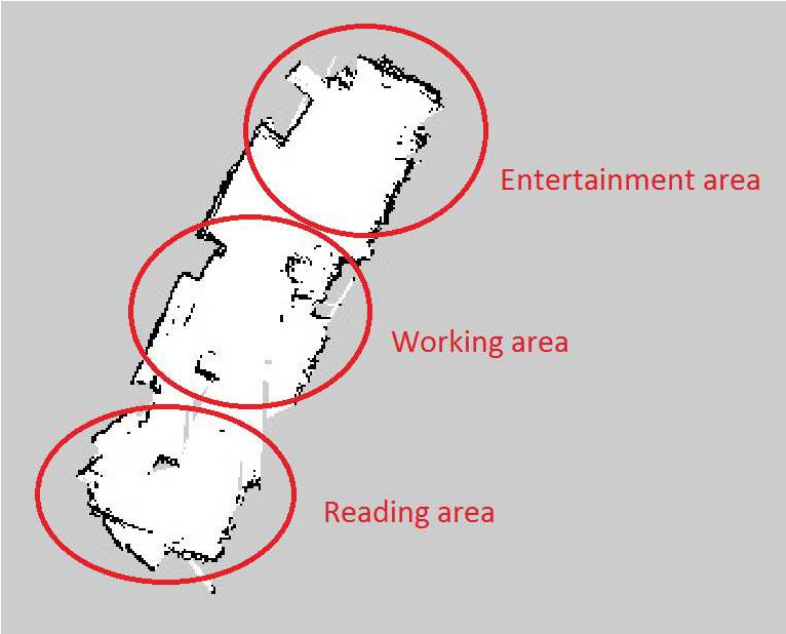


Figura 6.14: Mapa geométrico de la sala de pruebas.

6.3.1. Aprendizaje con diálogo

El sistema de aprendizaje con diálogo permite al sistema que aprenda mediante preguntas directas al usuario. Por esta vía, el robot puede aprender los objetos que sirven para una acción o utilidad concreta, así como las características que deben tener y el grado con el que se ajustan. Además permite que el robot aprenda de qué manera puede interactuar un objeto con otro y qué acciones se pueden asociar con un significado añadido concreto.

En esta sección se documenta una prueba de aprendizaje con diálogo mientras se probaba la integración con el sistema de interacción por voz. Este experimento es interesante porque **involucra el navegador geométrico, el planificador semántico, el razonador y el sistema de interacción con el usuario por voz**. Aunque esta sección está centrada en el aprendizaje por los resultados del sistema de diálogo.

Descripción del entorno y de las condiciones previas a la prueba

Esta prueba fue realizada en la sala de profesores del campus de Leganés de la Universidad Carlos III de Madrid, dicha sala se dividió en tres secciones diferentes basadas en los objetos que la contenían. El mapeado geométrico se realizó primero con una herramienta de SLAM (*Simultaneous Localization And Mapping*) incluida en ROS (gmapping). Y posteriormente fue introducido el etiquetado semántico de las estancias (ver figura 6.14), así como la información relativa a los objetos contenidos y su posición. La figura 6.15 es un dibujo esquemático del entorno donde se muestra además el primer camino que recorre el robot antes de decidir que necesita aprender del usuario.

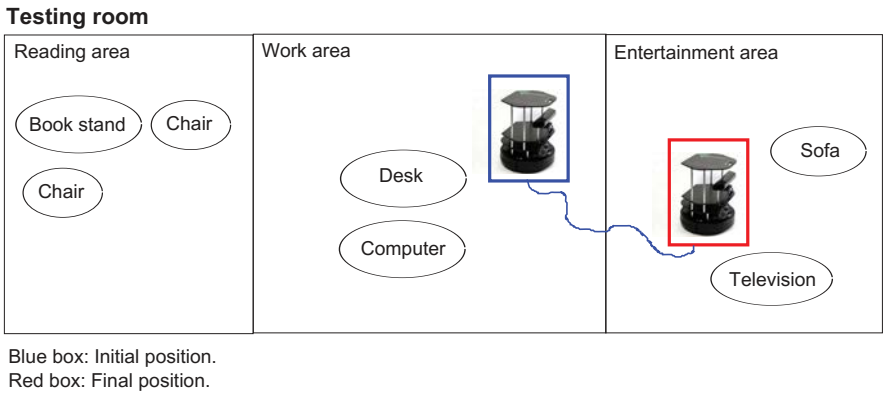


Figura 6.15: Mapa esquemático de la sala de pruebas

Se espera la siguiente secuencia:

- Solicitar ir a un lugar divertido.
- El robot va a la televisión porque tiene la información de que *ver la tele* es *divertido* y el televisor sirve para ver la tele. Es un objeto conocido (está en el mapa) y se dirige allí.
- El usuario no está de acuerdo.
- El robot se dirige a la sala de lectura porque tiene la información de que en la sala de lectura puede haber libros y los libros sirven para leer y leer es divertido.

Figura 6.16: Posición final de la trayectoria del robot: en frente de la televisión.

- El usuario no está conforme.
- El robot inicia el proceso de aprendizaje por preguntas al usuario.
- El robot se dirige a un lugar facilitado por la nueva información.

Descripción del proceso de la prueba

El robot se comportó siguiendo la secuencia esperada. Se dirigió a donde estaba el televisor, tal como indicaba la figura 6.15. La posición final que alcanza el robot en esta parte del experimento se puede ver en la figura 6.16. Después, ante la negativa del usuario, el robot se dirige a la sala de lectura como se muestra en la figura 6.17. Aunque no hubiera ningún libro identificado en el entorno, el razonador tenía la información de que en la sala de lectura es el mejor sitio para encontrar un libro.

Ante la segunda negativa es cuando el sistema responde iniciando el proceso de aprendizaje mediante diálogo. La secuencia resumida es así:

Robot: Comunica al usuario que no ha podido encontrar el destino y le solicita que le diga acciones que son divertidas.

Usuario: Responde que jugar es divertido.

Robot: Aprende (relaciona) que jugar es divertido, y conoce algunos objetos que sirven para jugar pero pregunta al usuario si le puede decir algún otro objeto.

Figura 6.17: Posición final de la trayectoria del robot: en una *zona de lectura*.

Usuario: Responde que el objeto ordenador puede servir para jugar.

Robot: Pregunta si todos los ordenadores o sólo algunos ordenadores concretos.

Usuario: Todos los ordenadores.

Robot: Pregunta que cómo se ajusta un ordenador a la utilidad jugar.

Usuario: Bastante bien.

Robot: Pregunta si el ordenador interacciona con algún objeto de los que ya conoce u otro nuevo.

Usuario: Desestima la posibilidad de afirmar eso y dice que no.

Robot: Pregunta al usuario si quiere repetir la búsqueda por algo divertido con la nueva información.

Usuario: Asiente.

Tras ese diálogo, el robot asocia *jugar* con *divertido* y *ordenador* con *jugar*. Puesto que el usuario quiere repetir la búsqueda, el razonador hace que el planificador le diga

al navegador geométrico que se dirija al lugar donde está el ordenador. El usuario entonces confirma el destino y se finaliza la navegación. Con esto se prueba que el robot es capaz de aprender del diálogo.

6.3.2. Etiquetado de estancias con sonido y detección de personas

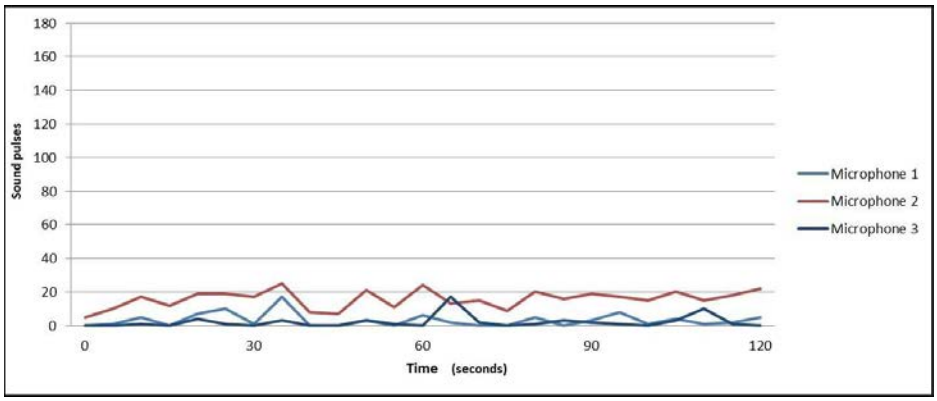
La capacidad de identificar y etiquetar tipos de estancia puede considerarse como una capacidad de aprendizaje autónomo puesto que el robot obtiene información por sí mismo sin intervención humana. El sistema ha sido dotado de la habilidad de etiquetado de estancias mediante el sonido ambiente y el movimiento que realizan las personas en esa estancia. En esta sección se describen los experimentos sobre esta habilidad. Los resultados se recogen en [32].

Descripción del experimento básico

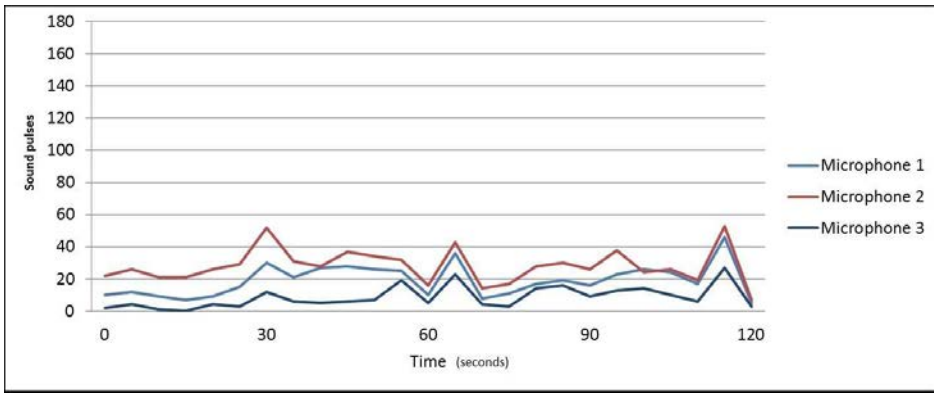
El sistema propuesto ha sido probado en seis escenarios diferentes: *cafetería*, *biblioteca*, *pasillo*, *sala de exposiciones*, *sala de conferencias* y *campo de fútbol interior*. Se ha tenido en cuenta el desplazamiento de cada persona detectada y el sonido ambiente. Los datos registrados han sido el número de personas detectadas simultáneamente, el desplazamiento total, la media del desplazamiento, la desviación típica del desplazamiento y los datos de sonido de tres micrófonos.

Respecto al escenario de la biblioteca (figura 6.20a), se tomaron 101 muestras. Los datos del sonido ambiente se muestran en la figura 6.18a, que con un entorno silencioso. En este experimento, los datos fueron tomados situando manualmente los sensores en distintos lugares del entorno.

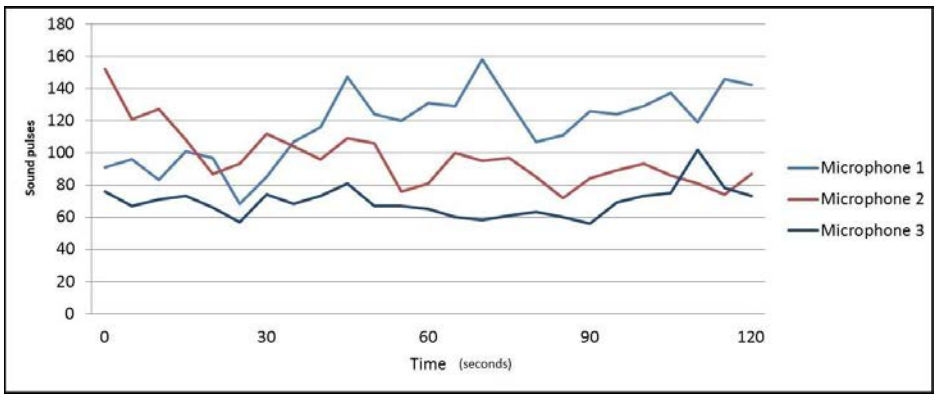
En el escenario del pasillo (figura 6.20d), se tomaron 100 muestras. La figura 6.18b muestra los datos obtenidos del sonido ambiente. Se tomaron la misma cantidad de muestras en el entorno de la cafetería, cuyos datos de sonido recogidos se muestran en la figura 6.18c. En la cafetería los datos fueron tomados por el propio robot equipado con el sistema sensorial completo (figura 6.19) mientras era teleoperado para alcanzar diferentes zonas del entorno, donde el robot tomaba cierta cantidad de muestras antes de ser llevado a otra zona. Una parte de las muestras también fueron tomadas situando



(a)



(b)



(c)

Figura 6.18: Muestras de sonido ambiente. (**a**) Sonido en la biblioteca; (**b**) Sonido en el pasillo; (**c**) Sonido en la cafetería.

Figura 6.19: Turtlebot equipado con micrófonos y el sensor RGB-D de Asus.

los sensores del robot en determinados puntos de la cafetería (figura 6.20b). En el pasillo el sistema sensorial fue ubicado manualmente en un punto seleccionado desde donde el robot percibía a las personas pasar.

La cafetería se considera un entorno potencialmente ruidoso, mientras que el sonido del pasillo es variable pero el desplazamiento de las personas se presupone mayor. Junto con la biblioteca, que se considera un entorno silencioso, se tienen los tres escenarios elegidos para las primeras pruebas de clasificación de lugares con este método.

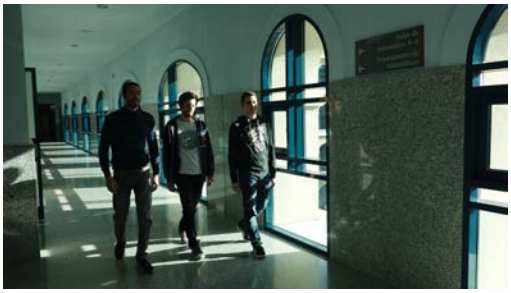
El número de personas detectado en cada entorno se muestra en el gráfico de la figura 6.22. Los datos sobre el desplazamiento de las personas detectadas en cada entorno se muestran en la figura 6.21. Los datos de sonido de los tres micrófonos de cada muestra fueron añadidos y divididos por el número de muestras. El resultado de la suma ha sido normalizado y se muestra en la figura 6.23. Mientras que la biblioteca y el pasillo cuentan con datos similares de sonido, una notable diferencia se observa en el entorno de la cafetería. Esto conlleva que diferenciar entre la cafetería y los otros dos entornos sea posible empleando únicamente los datos del sonido. Sin embargo la

(a)

(b)

(c)

(d)



(e)

Figura 6.20: Imágenes de los escenarios muestreados. (**a**) Biblioteca; (**b**) Cafetería; (**c**) Sala de exposiciones; (**d**) Pasillo; (**e**) Otro pasillo.

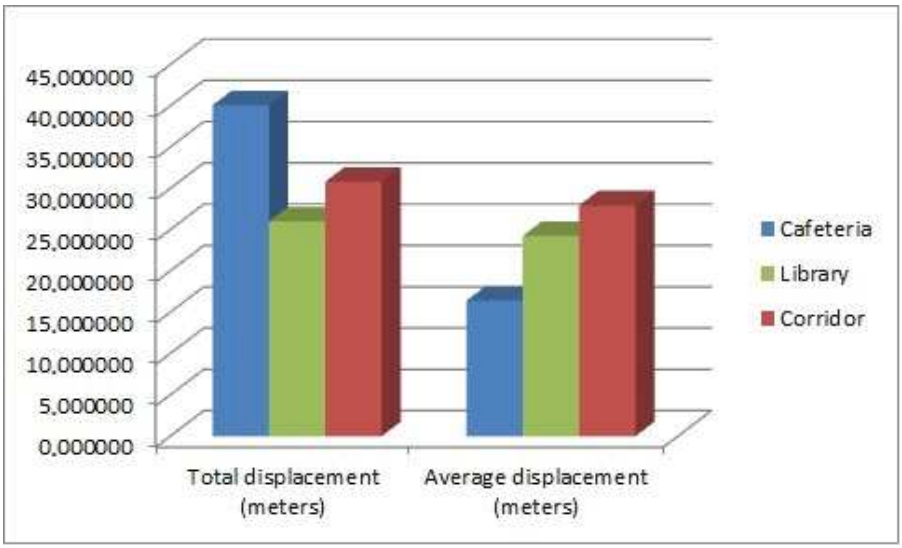


Figura 6.21: Muestras de desplazamiento.

inclusión de las observaciones extraídas del movimiento de las personas mejoran la tarea del etiquetado en situaciones menos evidentes y se comenta en experimentos añadidos más adelante.

Descripción de los experimentos avanzados

Tras la primera batería de pruebas, se han realizado algunos experimentos para esclarecer algunos aspectos del funcionamiento que no quedaban claros. Por ejemplo, quedaba la duda de si la combinación de las características del entorno, como el sonido y el movimiento de las personas, repercutía realmente en una mayor efectividad de la clasificación. Para ello se han tomado muestras en dos entornos cuyas características sonoras y de movimiento de personas son aparentemente diferentes y suficientes para clasificar los entornos, por lo que se han sometido a la SVM a una serie de pruebas para un estudio por ablación. Primero sin datos de sonido, luego en otra serie de pruebas sin datos de personas y luego una última prueba con todos los datos.

Estos dos entornos han sido el pasillo y un nuevo entorno definido como *sala de exposiciones* cuya imagen se puede ver en la figura 6.20c. A la hora de hacer las pruebas, se ha entrenado la SVM con el 70 % de las muestras tomadas aleatoriamente y se ha reservado el 30 % para el conjunto de test, como en los experimentos básicos.

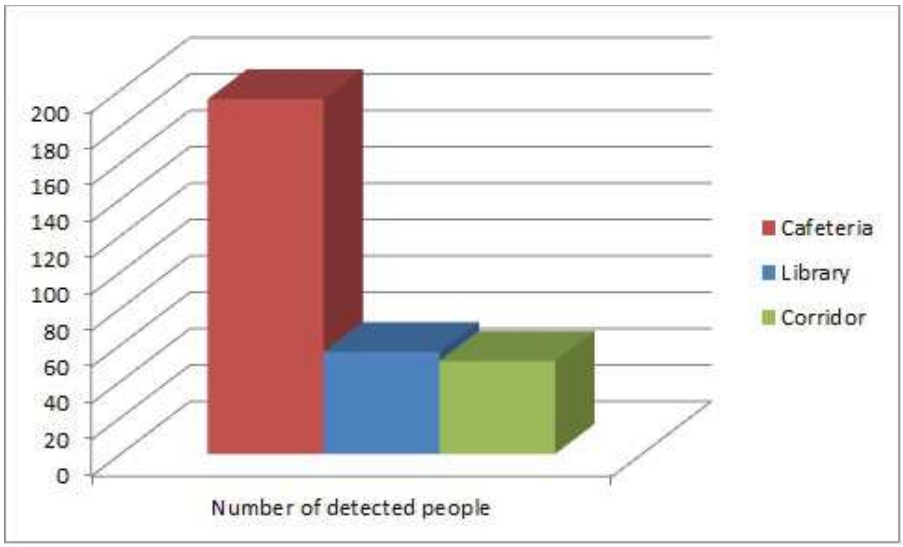


Figura 6.22: Personas detectadas.

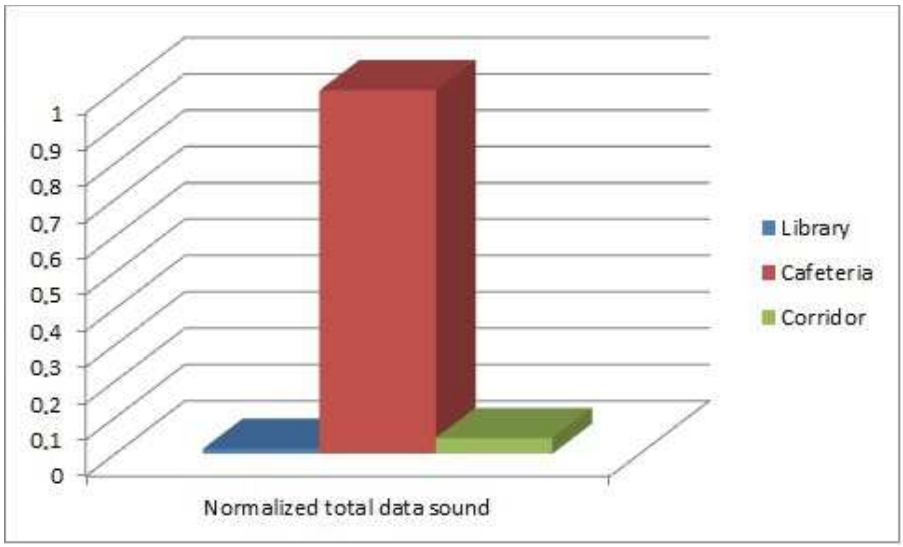


Figura 6.23: Datos de sonido de la cafetería, el pasillo y la biblioteca.

En esta sección se toman en cuenta los entornos que no se contemplaron en los experimentos básicos. Aparte de la sala de exposiciones se han incluído los entornos correspondientes a un campo de fútbol cubierto y una sala de conferencias donde se han dado charlas.

Otro aspecto sin comentar de la sección de experimentos previa es la capacidad de identificar estancias cuyas muestras no han sido utilizadas para entrenar la SVM. Para comprobar este asunto, se han tomado muestras de pasillo en un lugar distinto, mostrado en la figura 6.20e. Y se ha realizado una prueba entrenando una SVM con todas las muestras del pasillo de los experimentos básicos. En el conjunto de entrenamiento también se han añadido las muestras de la cafetería, para ver si la SVM entrenada con todas las muestras era capaz de identificar como pasillo las muestras tomadas en el pasillo nuevo.

Análisis de los resultados de los experimentos básicos

Para evaluar la validez del etiquetado semántico en función de la información del sonido ambiental y el movimiento de las personas detectadas, los resultados han sido presentados en matrices de confusión. Se ha estudiado cada caso particular. Los clasificadores han sido creados con SVM, recogiendo todas las muestras tomadas en un tipo de habitación. Luego se entrena el clasificador con el 70 % de esas muestras, elegidas aleatoriamente, y se reserva el 30 % restante de esas muestras para testear el clasificador. Se ha repetido el proceso doce veces por cada clasificador y los resultados han sido estudiados.

- **Biblioteca vs Cafetería**. El primer objetivo es comprobar que un sistema robótico entrenado puede diferenciar tipos de estancia con la información sensorial disponible, pero centrada en la capacidad de deducir lo que realizan las personas en ese lugar. Deducir exactamente qué están haciendo las personas es difícil, pero darse cuenta de que las personas realizan acciones distintas en distintos lugares, es más sencillo. El primer experimento ha sido comprobar que lugares muy distintos son bien etiquetados. Los lugares que se han elegido son Biblioteca y Cafetería. Son lugares donde la gente hace cosas muy diferentes, y queda plasmado en lo que el sistema mide: nivel sonoro y cantidad de movimiento de las personas. En principio, una cafetería es más ruidosa y se aglomera más

cantidad de gente por metro cuadrado. En una biblioteca el sonido ambiente es mucho menor, tiende a haber más espacio libre entre la gente y el movimiento es menor. Aún cuando una persona pasa por delante del sensor, se estima que su velocidad será menor. La eficacia diferenciando estos dos tipos de estancias ha sido muy buena. Esto queda reflejado en la tabla 6.12. Esta tabla muestra los resultados de uno de los 12 clasificadores generados. La tabla 6.13 es el resultado de sumar los resultados de los 12 clasificadores. Aunque los experimentos arrojaron algunos clasificadores mejores que otros, todos ofrecían muy buenos resultados.

La tasa de estancias correctamente etiquetadas en este experimento fue del 98.85 %. Es un buen resultado, tal como se esperaba en este primer caso.

- **Biblioteca vs Pasillo**. El segundo objetivo es poner un caso difícil al sistema. El pasillo también puede ser bastante silencioso y las muestras se tomaron en un momento de poco tránsito. Además, algunas personas en la biblioteca simplemente caminaron delante del sensor, lo que es la misma acción que realizaban en el pasillo. Por último, por limitaciones sensoriales, el sistema encontraba dificultades en identificar personas sentadas en la biblioteca a cierta distancia (el sensor ASUS está pensado para distancias de 3 metros y el algoritmo de detección de personas se centra en las piernas). Por todo ello, se considera un experimento desafiante. El resultado, sin embargo, es mejor de lo esperado. Algunos de los 12 clasificadores generados, ofrecían buenos resultados como se puede ver en la tabla 6.17. Esta tabla muestra uno de los 12 clasificadores que ofrecía buenos resultados. Si se tienen en cuenta más clasificadores de los que ofrecían un buen porcentaje de éxitos, los resultados son similares como se puede ver en la tabla 6.18. El ratio de éxito en esta prueba alcanza el 91.5 %, lo que se considera una cifra muy buena, sólo el 13.2 % de las muestras del pasillo fueron clasificadas como *biblioteca*. La suma de todos los clasificadores, tanto los buenos como los malos, ofrece un resultado que se muestra en la tabla 6.19. El ratio de éxito cuando se incluyen también los peores clasificadores obtenidos es del 86.9 %.

	Etiqueta Detectada		
Etiqueta Real	Biblioteca	Cafetería	Total
Biblioteca	35 (97.22 %)	1 (2.77 %)	36
Cafetería	1 (3.7 %)	26 (96.29 %)	27

Tabla 6.12: Tabla con un clasificador para diferenciar entre Biblioteca y Cafetería.

	Etiqueta Detectada		
Etiqueta Real	Biblioteca	Cafetería	Total
Biblioteca	315 (98.43 %)	5 (1.56 %)	320
Cafetería	2 (0.69 %)	286 (99.3 %)	288

Tabla 6.13: Tabla con los resultados de tener en cuenta los doce clasificadores generados para diferenciar entre Biblioteca y Cafetería.

- **Biblioteca vs Cafetería vs Pasillo**. El siguiente objetivo es ver la eficacia diferenciando varios tipos de estancia a la vez. Se han combinado los casos sencillos y complicados para diferenciar entre Biblioteca, Cafetería y Pasillo. Tomando un buen clasificador de entre los 12 clasificadores generados, los resultados son los de la tabla 6.14. Combinando varios buenos clasificadores, se obtiene la tabla 6.16. La suma de todos los clasificadores generados se muestra en la tabla 6.15, la cual muestra una tasa de acierto para estancias clasificadas correctamente del 91.6 %

Análisis de los resultados de los experimentos avanzados

En la sección 6.3.2 se ha comentado que se llevó a cabo otra serie de pruebas para obtener más resultados sobre el funcionamiento de este sistema.

	Etiqueta Detectada			
Etiqueta Real	Biblioteca	Cafetería	Pasillo	Total
Biblioteca	31 (96.87 %)	0 (0 %)	1 (3.12 %)	32
Cafetería	0 (0 %)	33 (97.06 %)	1 (2.94 %)	34
Pasillo	2 (10 %)	0 (0 %)	18 (90 %)	20

Tabla 6.14: Tabla con un buen clasificador obtenido para diferenciar entre Biblioteca, Cafetería y Pasillo

	Etiqueta Detectada			
Etiqueta Real	Biblioteca	Cafetería	Pasillo	Total
Biblioteca	301 **(93.19 %)**	9 (2.78 %)	13 (4.02 %)	323
Cafetería	1 (0.26 %)	378 **(99.21 %)**	2 (0.52 %)	381
Pasillo	60 (18.99 %)	0 (0 %)	256 **(81.01 %)**	316

Tabla 6.15: Tabla con la suma de los doce clasificadores generados para diferenciar entre Biblioteca, Cafetería y Pasillo

	Etiqueta Detectada			
Etiqueta Real	Biblioteca	Cafetería	Pasillo	Total
Biblioteca	103 (92.79 %)	3 (2.7 %)	5 (4.5 %)	111
Cafetería	0 (0 %)	133 (99.25 %)	1 (0.75 %)	134
Pasillo	13 (13 %)	0 (0 %)	87 (87 %)	100

Tabla 6.16: Tabla con varios buenos clasificadores para diferenciar entre Biblioteca, Cafetería y Pasillo

	Etiqueta Detectada		
Etiqueta Real	Biblioteca	Pasillo	Total
Biblioteca	22 (95.65 %)	1 (4.35 %)	23
Pasillo	4 (12.9 %)	27 (87.1 %)	31

Tabla 6.17: Tabla con un buen clasificador obtenido para diferenciar entre Biblioteca y Pasillo

	Etiqueta Detectada		
Etiqueta Real	Biblioteca	Pasillo	Total
Biblioteca	51 (96.23 %)	2 (3.77 %)	53
Pasillo	7 (13.2 %)	46 (86.8 %)	53

Tabla 6.18: Tabla considerando varios buenos clasificadores

	Etiqueta Detectada		
Etiqueta Real	Biblioteca	Pasillo	Total
Biblioteca	305 (96.21 %)	12 (3.78 %)	317
Pasillo	72 (22.15 %)	253 (77.85 %)	325

Tabla 6.19: Tabla con el resultado de tener en cuenta los doce clasificadores del experimento

	Etiqueta Detectada		
Etiqueta Real	**Expo**	**Pasillo**	**Total**
Expo	207 (75.8 %)	66 (24.2 %)	273
Pasillo	86 (29.1 %)	209 (70.8 %)	295

Tabla 6.20: Resultados obtenidos con la suma de los diez clasificadores para diferenciar entre pasillo y sala de exposiciones, **sin los datos de sonido ambiente**.

- **Estudio por ablación**: Los entornos elegidos para esta prueba son un pasillo y la sala de exposiciones. La tabla 6.20 muestra el resultado de las pruebas de clasificación cuando se eliminan los datos de sonido. Como se puede apreciar, el ratio de clasificación ofrece un resultado aceptable pero mejorable. La tabla 6.21 muestra el resultado de la clasificación cuando se eliminan los datos relativos a las personas y su movimiento. El ratio es peor que en el caso anterior, sobre todo tratando de clasificar la estancia *sala de exposición*. En cualquier caso, cuando se combinan todos los datos, el ratio sube considerablemente, tal como muestra la tabla 6.22.

- **Pruebas con más entornos**. Se ha realizado una batería de pruebas generando diez clasificadores a partir de las muestras tomadas en los entornos *sala de exposiciones*, *campo de fútbol cubierto*, *sala de conferencias*, *biblioteca*, *cafetería* y *pasillo*. La tabla 6.23 recoge la suma de los diez clasificadores generados.

- **Pruebas con entornos diferentes a los de entrenamiento**. Se ha entrenado una SVM con todas las muestras de la cafetería y del pasillo de los test básicos. En este experimento sólo se puede generar un clasificador, puesto que al tomar el 100 % de las muestras de ambos entornos para el entrenamiento, no hay combinaciones aleatorias. El test se ha realizado con el 100 % de las muestras tomadas en el segundo pasillo. En total 93 muestras de pasillo para el test. El resultado es el de la tabla 6.24.

	Etiqueta Detectada		
Etiqueta Real	**Expo**	**Pasillo**	**Total**
Expo	156 (48.3 %)	167 (51.7 %)	323
Pasillo	42 (16.2 %)	217 (83.7 %)	259

Tabla 6.21: Resultados obtenidos con la suma de diez clasificadores para diferenciar entre pasillo y sala de exposiciones, **sin los datos de movimiento de personas**.

	Etiqueta Detectada		
Etiqueta Real	**Expo**	**Pasillo**	**Total**
Expo	243 (80.2 %)	60 (19.8 %)	303
Pasillo	64 (23.2 %)	211 (76.8 %)	275

Tabla 6.22: Resultados obtenidos con la suma de diez clasificadores para diferenciar entre pasillo y sala de exposiciones, empleando todos los datos completos.

	Etiqueta Detectada						
E. Real	Expo	Fútbol	Conf.	Biblio.	Cafetería	Pasillo	Tot.
Expo	269 (92.4 %)	0 (0 %)	0 (0 %)	20 (6.8 %)	0 (0 %)	2 (0.6 %)	291
Fútbol	0 (0 %)	288 (92.6 %)	11 (3.5 %)	7 (2.1 %)	19 (5.7 %)	6 (1.8 %)	331
Conf.	8 (3.5 %)	13 (5.7 %)	164 (72.2 %)	10 (4.4 %)	17 (7.5 %)	15 (6.6 %)	227
Biblio.	17 (5.4 %)	2 (0.6 %)	0 (0 %)	289 (91.7 %)	1 (0.3 %)	6 (1.9 %)	315
Cafetería	1 (0.3 %)	43 (14.8 %)	47 (16 %)	0 (0 %)	202 (69 %)	0 (0 %)	293
Pasillo	3 (1.2 %)	1 (0.4 %)	0 (0 %)	42 (17.7 %)	0 (0 %)	191 (80.6 %)	237

Tabla 6.23: Resultados obtenidos empleando varios buenos clasificadores para diferenciar todos los escenarios.

	Etiqueta Detectada		
Etiqueta Real	**Pasillo**	**No Pasillo**	**Total**
Pasillo	70 (75.26 %)	23 (24.73 %)	93

Tabla 6.24: Resultados obtenidos de probar un nuevo escenario de pasillo con los datos del anterior.

Los test avanzados permiten comprobar que hay entornos fácilmente identificables con el sistema propuesto y que la fusión de las variables tenidas en cuenta en esta

sección puede mejorar la identificación que se podría realizar con las variables por separado. Además, la identificación de un lugar cuyas muestras no han sido incluidas en el conjunto de entrenamiento ha tenido más del 75 % de éxito. Se observa que en algunas situaciones el sistema puede confundir lo que están haciendo las personas y provocar fallos en la clasificación, como por ejemplo en el entorno de la cafetería se aprecia que los sensores detectaban similitudes en el movimiento y sonido de las personas con los entornos *Fútbol* y *Conferencia*. Esto probablemente es debido a que hay muestras en la cafetería con personas muy quietas (como en la conferencia) y muestras en las que hay bastante movimiento y bastante cantidad de gente (como en el campo de fútbol). Además, los datos de sonido también varían mucho en los tres escenarios. Se supone que mejorando el sistema sensorial para incluir más información de la gente, como reconocimiento de expresiones faciales, estos fallos se reducirán.

Análisis global

El sistema permite etiquetar correctamente tipos de estancias distintas, porque detecta que se están realizando acciones diferentes. El supuesto de poder ampliar mecanismos de etiquetado semántico de localizaciones, en base a lo que hacen las personas en dichas localizaciones, ha quedado confirmado. Los resultados son mejores cuanto más distintas son las características que se han tomado. Como trabajo futuro, se plantea mejorar el sistema sensorial. Estas mejoras pueden consistir en añadir un laser para la detección de personas, un algoritmo de detección de caras y mejores micrófonos. Esto permitirá añadir nuevos atributos a tener en cuenta, como la disposición en el espacio de personas hablando y las expresiones faciales de los individuos que pueblan el entorno.

Es notorio que dependiendo de la hora del día, estas circunstancias pueden cambiar. Sin embargo, este sistema no pretende sustituir otros sistemas de etiquetado semántico basados en otros elementos fijos del entorno, sino que este sistema está orientado a complementar y ampliar otros sistemas. Si se utiliza de modo independiente, habría que tener en cuenta que el etiquetado que asigne el robot a una estancia será dinámico y variará en función de lo que haga la gente en ese momento. Esta característica dinámica se considera positiva por ofrecer una alternativa a otros métodos de etiquetado ya existentes. El enfoque sigue siendo interesante, el mismo lugar puede

ser etiquetado de manera diferente en función de lo que esté haciendo la gente en ese momento.

6.3.3. Pruebas de predicción de destino

En esta sección se describen las pruebas de predicción de destino. Es conveniente recordar que estas pruebas únicamente pretenden demostrar la posible viabilidad de este sistema para trabajos futuros. La elección de atributos útiles para construir estos árboles de decisión es un tema abierto.

Las pruebas han consistido en crear un fichero de muestras sobre el que se han ido construyendo árboles de decisión aleatorios. A partir de estos árboles se han extraído reglas que luego han permitido deducir el destino a partir de una serie de preguntas.

Datos de entrenamiento

Para validar el sistema ha sido necesario partir de datos de entrenamiento figurados, no han sido basados aún en ningún estudio. En las pruebas realizadas únicamente se pretendía comprobar la viabilidad de este método. En trabajos futuros se someterá a unas pruebas con casos reales para comprobar la utilidad real de este sistema.

Se han creado 17 muestras que se pueden ver en la captura del archivo de la figura 6.24.

Reglas generadas

En esta sección se muestran y comentan algunas de las reglas generadas con los datos de la figura 6.24. Pese a que esta línea de investigación se ha quedado en una primera aproximación, los resultados previos muestran interesantes conclusiones. Por ejemplo, en la figura 6.25 se muestra el set de reglas del conjunto de sets de reglas generadas que requiere pocas preguntas al usuario para predecir el destino. Llama la atención que esta situación se obtiene cuando el primer atributo a analizar es el del objeto requerido. En ese caso, las preguntas son muy breves y directas. Este conjunto de reglas consta de diez elementos.

Otro set de reglas de diez elementos pueden verse en la figura 6.26. Pero otro resultado interesante es el de los sets de reglas de la figura 6.27 puesto que se reduce

```
"Navegacion robots"
Accion Descansar Comer Jugar Leer Dormir Beber Trabajar *
TranquilidadNecesaria Mucha Media Poca *
TiempoNecesario Mucho Medio Poco *
Urgencia Mucha Media Poca *
Objeto Cama Silla Consola Comida Tele Libro Ordenador Frigorifico Agua Sofa Refresco *
Animo Contento Neutro Triste *
Hambriento Si No *
Cansado Nada Bastante Mucho *
Nervioso Si No *
Emocionado Si No *
Temperatura Calor Normal Frio *
EntornoRuidoso Si No *
LuminosidadNecesaria Alta Media Baja *
CantidadGente Poca Mucha *
Estancia Comedor Salon Cocina Dormitorio Despacho Aseo Terraza *
(x1  Descansar Mucha Mucho Media Cama Neutro No Mucho No No Normal No Baja Poca Dormitorio)
(x2  Descansar Media Poco Mucha Silla Neutro No Mucho No No Normal No Media Mucha Salon)
(x3  Comer Media Medio Mucha Comida Neutro Si Nada No No Normal No Alta Mucha Comedor)
(x4  Comer Poca Poco Mucha Comida Neutro Si Nada No No Normal No Alta Poca Cocina)
(x5  Jugar Poca Medio Poca Consola Contento No Nada No No Normal Si Media Mucha Salon)
(x6  Leer Mucha Medio Poca Libro Neutro No Nada No No Normal No Alta Poca Salon)
(x7  Trabajar Mucha Mucho Media Ordenador Neutro No Nada No No Normal No Alta Poca Despacho)
(x8  Jugar Poca Medio Poca Ordenador Contento No Nada No No Normal No Media Poca Despacho)
(x9  Beber Poca Poco Mucha Agua Neutro No Nada No No Normal No Alta Poca Cocina)
(x10  Comer Poca Poco Mucha Frigorifico Neutro Si Nada No No Normal No Alta Mucha Cocina)
(x11  Beber Poca Poco Mucha Agua Neutro No Bastante No No Normal No Alta Mucha Cocina)
(x12  Beber Poca Poco Mucha Agua Neutro No Nada Si No Normal No Alta Poca Cocina)
(x13  Beber Poca Poco Mucha Agua Neutro No Nada No No Normal Si Alta Mucha Cocina)
(x14  Beber Poca Poco Mucha Refresco Neutro No Nada No No Normal No Alta Poca Cocina)
(x15  Beber Poca Poco Mucha Agua Neutro No Nada No No Calor No Alta Poca Cocina)
(x16  Jugar Poca Medio Poca Ordenador Contento No Nada No No Calor No Media Mucha Despacho)
(x17  Jugar Poca Medio Poca Ordenador Contento No Nada No No Frio No Media Poca Despacho)
*
```

Figura 6.24: Captura del archivo de muestras utilizado para las pruebas de creación de árboles de decisión.

```
((REGLA CF 1.0 IF ((OBJETO ES ORDENADOR)) THEN ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF ((OBJETO ES AGUA)) THEN ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF ((OBJETO ES REFRESCO)) THEN ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF ((OBJETO ES FRIGORIFICO)) THEN ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF ((OBJETO ES LIBRO)) THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF ((OBJETO ES CONSOLA)) THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF
  ((CANTIDADGENTE ES POCA) (ENTORNORUIDOSO ES NO) (OBJETO ES COMIDA)) THEN
  ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF
  ((CANTIDADGENTE ES MUCHA) (ENTORNORUIDOSO ES NO) (OBJETO ES COMIDA)) THEN
  ((ESTANCIA COMEDOR)))
 (REGLA CF 1.0 IF ((OBJETO ES SILLA)) THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF ((OBJETO ES CAMA)) THEN ((ESTANCIA DORMITORIO))))
```

Figura 6.25: Set de reglas que comienzan con el atributo de objeto requerido.

```
((REGLA CF 1.0 IF
  ((CANTIDADGENTE ES POCA) (URGENCIA ES POCA) (ACCION ES JUGAR)) THEN
  ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF
  ((ENTORNORUIDOSO ES SI) (CANTIDADGENTE ES MUCHA) (URGENCIA ES POCA)
   (ACCION ES JUGAR))
  THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF
  ((ENTORNORUIDOSO ES NO) (CANTIDADGENTE ES MUCHA) (URGENCIA ES POCA)
   (ACCION ES JUGAR))
  THEN ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF ((ACCION ES BEBER)) THEN ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF ((TIEMPONECESARIO ES MEDIO) (ACCION ES COMER)) THEN
  ((ESTANCIA COMEDOR)))
 (REGLA CF 1.0 IF ((TIEMPONECESARIO ES POCO) (ACCION ES COMER)) THEN
  ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF ((ACCION ES TRABAJAR)) THEN ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF ((ACCION ES LEER)) THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF ((URGENCIA ES MEDIA) (ACCION ES DESCANSAR)) THEN
  ((ESTANCIA DORMITORIO)))
 (REGLA CF 1.0 IF ((URGENCIA ES MUCHA) (ACCION ES DESCANSAR)) THEN
  ((ESTANCIA SALON))))
```

Figura 6.26: Set de reglas de diez elementos.

el número de elementos necesarios para abarcar todos los casos de las muestras. En estos sets de reglas, ocho reglas son suficientes, lo que indica que las distintas combinaciones de reglas en función del atributo que se elige para empezar a construir el árbol, pueden ofrecer características diferentes.

La conclusión que se puede extraer de estos experimentos es que con suficiente información sobre las circunstancias que motivan a un usuario para ir a un determinado lugar, se puede minimizar algunos de los aspectos de esta técnica de predicción de destinos. Si interesa que las preguntas sean menos frecuentes, las reglas utilizadas deben iniciarse con el atributo más discriminatorio. En estos ejemplos, con el objeto que el usuario quiere utilizar. Sin embargo se pueden generar árboles de decisión que lleven a un menor número de reglas, que puede ser otro factor interesante a minimizar por temas de rendimiento. Estos resultados previos indican que existe un margen de ajuste optimizable.

```
((REGLA CF 1.0 IF
  ((TRANQUILIDADNECESARIA ES MEDIA) (LUMINOSIDADNECESARIA ES MEDIA)) THEN
  ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF
  ((OBJETO ES CONSOLA) (CANSADO ES NADA) (EMOCIONADO ES NO) (NERVIOSO ES NO)
   (TRANQUILIDADNECESARIA ES POCA) (LUMINOSIDADNECESARIA ES MEDIA))
  THEN ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF
  ((OBJETO ES ORDENADOR) (CANSADO ES NADA) (EMOCIONADO ES NO) (NERVIOSO ES NO)
   (TRANQUILIDADNECESARIA ES POCA) (LUMINOSIDADNECESARIA ES MEDIA))
  THEN ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF
  ((TIEMPONECESARIO ES POCO) (URGENCIA ES MUCHA)
   (LUMINOSIDADNECESARIA ES ALTA))
  THEN ((ESTANCIA COCINA)))
 (REGLA CF 1.0 IF
  ((TIEMPONECESARIO ES MEDIO) (URGENCIA ES MUCHA)
   (LUMINOSIDADNECESARIA ES ALTA))
  THEN ((ESTANCIA COMEDOR)))
 (REGLA CF 1.0 IF ((URGENCIA ES POCA) (LUMINOSIDADNECESARIA ES ALTA)) THEN
  ((ESTANCIA SALON)))
 (REGLA CF 1.0 IF ((URGENCIA ES MEDIA) (LUMINOSIDADNECESARIA ES ALTA)) THEN
  ((ESTANCIA DESPACHO)))
 (REGLA CF 1.0 IF ((LUMINOSIDADNECESARIA ES BAJA)) THEN
  ((ESTANCIA DORMITORIO))))
```

Figura 6.27: Set de reglas de ocho elementos.

Figura 6.28: Panorámica del escenario de pruebas de exploración.

6.4. Pruebas de exploración

El explorador semántico es el encargado de ubicar los objetos detectados en el mapa del navegador de bajo nivel. Para comprobar su funcionamiento se realizaron pruebas en el mismo laboratorio donde se habían realizado los experimentos con el sistema de clasificiación de objetos mediante Máquinas de Soporte Vectorial (SVM) [55] y se emplearon los mismos objetos para los que ese sistema de visión había sido entrenado. Además, en estas pruebas se consideró también la percepción por etiquetas, de este modo se hace una exploración que involucra la agregación de distintas tecnologías de visión que se han utilizado a lo largo de esta tesis.

El entorno de pruebas es un laboratorio en una carpa, acotado por una lona azul dentro de una nave del edificio Betancourt de la Universidad Carlos III de Madrid. Tiene una base rectangular con dos hileras de mesas y sillas para varios puestos de trabajo. También tiene tres armarios grandes, dos en el centro y otro en una esquina del lado opuesto. La figura 6.28 muestra una panorámica de 180º tomada desde la pared oeste de la carpa. Los objetos susceptibles de ser detectados por el robot son:

- **Armarios**. Deberían ser detectados por el sistema de identificación de objetos empleando SVM. En el entorno hay tres armarios.
- **Sillas**. Deberían ser detectadas por el sistema de identificación de objetos empleando SVM. En el entorno hay varias sillas cuya posición pudo variar de un experimento a otro.
- **Mesa**. Las mesas deberían ser detectadas por el sistema de identificación de etiquetas basado en artoolkit. Se asignó el identificador de un tipo de etiqueta con el concepto *mesa*.
- **Ordenador**. Para detectar ordenadores también se empleó el sistema de identificación de etiquetas. Inicialmente no se situó ninguna etiqueta asociada con el concepto *ordenador*. Se emplea en el experimento de identificación de estancia (sección 6.4.2).

Además, el explorador semántico es el encargado de identificar y clasificar la estancia en la que se encuentra en función de los objetos que percibe.

Para la realización de los experimentos de exploración, en la máquina del ordenador turtlebot estaban ejecutándose los siguientes launchers:

- *minimal.launch*, del paquete turtlebot_bringup. Corresponde a los drivers mínimos que deben estar activos para que el turtlebot se mueva. También lanza la odometría.
- *gmapping_demo.launch*, del paquete turtlebot_navigation. Corresponde al algoritmo de creación de mapas proporcionado de forma abierta en la comunidad ROS del turtlebot. Incluye los nodos de visión puesto que se emplea la información de profundidad del sensor de visión (el Xtion Pro de Asus) para simular el comportamiento de un láser.

- *ar_pose_multi_no_rviz.launch*, del paquete ar_pose. Es una modificación del launcher que venía por defecto en este paquete. Lanza los nodos necesarios para la detección de etiquetas usando ARToolkit.

En estos experimentos se consideró apropiado utilizar un segundo ordenador embarcado en la base dedicado exclusivamente al sistema de detección de objetos empleando SVM. A este ordenador se le conectó otro sensor de visión también en exclusividad. Los dos nodos que estaban activos eran:

- *openni_config.launch*, que es una adaptación del lanzador de nodo openni.launch, variando la configuración.

- *detection_node*, el nodo de detección de objetos cuyo funcionamiento fue publicado en [55].

En la estación de trabajo se lanzaron los siguientes nodos:

- *consultor_server*, del paquete semantic_navigation. Es el nodo que interactúa con la base de datos y ofrece servicios de consulta.

- *perception_aggregator*, del paquete semantic_navigation. El nodo que agrega y combina los diferentes sistemas de percepción.

- *reasoning_module*, del paquete semantic_navigation. El nodo que planifica y controla las inferencias del navegador. Necesario para comprobar cada vez que se detecta un objeto si ese objeto identifica la estancia.

- *explore_module*, del paquete semantic_navigation. Es el nodo que se pretende probar. En este experimento es lanzado en modo navegación geométrica. Ubica los objetos percibidos en el mapa y usa el módulo de razonamiento para clasificar la estancia.

- *keyboard_teleop.launch*, del paquete turtlebot_teleoperation. Se lanza para mover al robot manualmente por el entorno.

6.4.1. Identificación de objetos en el mapa

En este experimento se puso a prueba el explorador semántico mientras se construía el mapa de un navegador geométrico. El objetivo era comprobar que los objetos detectados por el sistema de percepción eran ubicados correctamente en el mapa. Aunque el robot tiene implementada una navegación de deambulación automática llamada *wandering*, en esta ocasión fue teleoperado manualmente para que recorriera todo el entorno. La exploración era detenida cuando el robot daba entre dos y cinco vueltas completas al entorno. El parámetro de la exploración relativos a la diferenciación de objetos ya identificados respecto a los objetos nuevos (ver la sección 3.3.3) fue establecido con el siguiente valor:

- MARGEN_SIMILITUD_COORDENADA: 0,6. Esto quiere decir que si se percibe un objeto a menos de 60 cm de un objeto igual anteriormente percibido, se considera que es el mismo objeto.

Los problemas identificados con estos experimentos eran originados sobre todo por falsos positivos provenientes del sistema de percepción. Pese a las medidas de precaución que se tomaron al desarrollar el agregador de percepciones (descrito en la sección 3.4.1), las limitaciones del sistema sensorial no pudieron ser eliminadas completamente. Dependiendo de la luz, los reflejos y demás circunstancias ambientales o de las características de los objetos, el sistema creía ver algo donde no había nada. Esto sucedía tanto con el sistema de detección de objetos usando SVM como con la detección de etiquetas de ARToolkit.

A continuación se muestran los objetos que detectó el robot en algunas ejecuciones. En primer lugar, el mapa creado por el navegador geométrico es el de la figura 6.31. En la imagen 6.29 vemos los objetos identificados por el sistema de percepción que fueron ubicados por el explorador. La figura 6.30 es un mapa esquemático representando dichas coordenadas. Los falsos positivos están marcados con un círculo rojo. En este caso, el sistema identificó razonablemente bien los tres armarios que había en el entorno. Sin embargo la etiqueta de la mesa la vió en sitios donde no debería estar. Y lo mismo sucedió con el contorno de la silla. Curiosamente falló al reconocer las sillas de verdad, pero es un error asumible pues el resto de experimentos muestran que las puede detectar, por lo tanto sólo sería necesario más tiempo de exploración.

```
mysql> select * from objeto_fisico;
+----------+--------------------+-------------------------+-----------------+----------+----------+-----------------+
| nombre   | nombre_conceptual  | nombre_caracteristica   | nombre_estancia | posicion | id_nodo  | numero_aciertos |
+----------+--------------------+-------------------------+-----------------+----------+----------+-----------------+
| chair-0  | chair              | NULO                    | estancia-1      |       67 |        0 |               0 |
| closet-0 | closet             | NULO                    | estancia-1      |       65 |        0 |               0 |
| closet-1 | closet             | NULO                    | estancia-1      |       66 |        0 |               0 |
| closet-2 | closet             | NULO                    | estancia-1      |       69 |        0 |               0 |
| TABLE-0  | TABLE              | NULO                    | estancia-1      |       68 |        0 |               0 |
| TABLE-1  | TABLE              | NULO                    | estancia-1      |       70 |        0 |               0 |
| TABLE-2  | TABLE              | NULO                    | estancia-1      |       71 |        0 |               0 |
+----------+--------------------+-------------------------+-----------------+----------+----------+-----------------+
7 rows in set (0.00 sec)

mysql> select * from posiciones_objeto_robot;
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
| id | posicion_obj_x | posicion_obj_y | posicion_obj_z | posicion_rob_x | posicion_rob_y | orientacion_rob_w | orientacion_rob_z |
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
| 65 |          2.34 |         -0.48 |         -1.00 |          1.37 |         -0.28 |            0.9729 |            0.2313 |
| 66 |          2.74 |          0.43 |         -1.00 |          1.31 |         -0.32 |            0.2260 |            0.9740 |
| 67 |          2.90 |          1.10 |         -1.00 |          1.37 |         -0.28 |            0.4000 |            0.9160 |
| 68 |         -0.12 |         -1.10 |         -1.00 |          0.63 |          0.24 |            0.3520 |           -0.9360 |
| 69 |         -1.68 |          2.69 |         -1.00 |          0.09 |          0.50 |            0.5093 |            0.8606 |
| 70 |          0.65 |         -0.93 |         -1.00 |          0.63 |          0.24 |           -0.8870 |            0.4620 |
| 71 |          1.77 |         -0.34 |         -1.00 |          0.70 |          0.13 |           -0.4980 |            0.8670 |
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
7 rows in set (0.00 sec)
```

Figura 6.29: Prueba 1 de localización de objetos en el mapa.

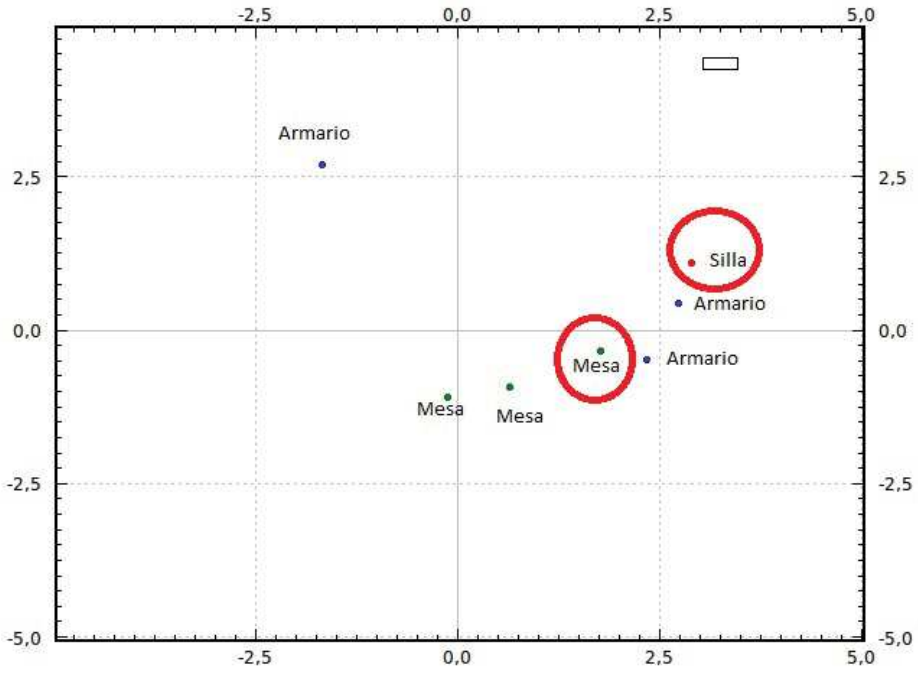


Figura 6.30: Mapa esquemático de los objetos localizados en la prueba 1.

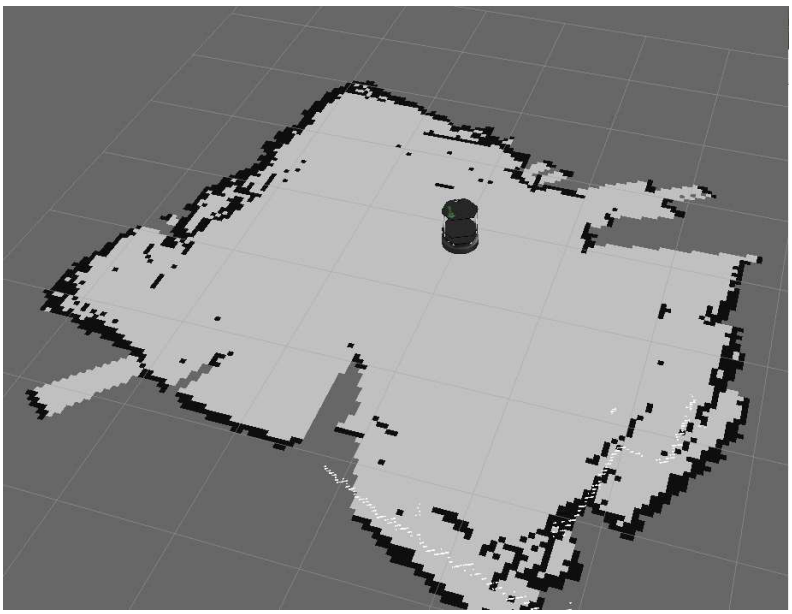


Figura 6.31: Mapa geométrico obtenido con gmapping mientras se realizaba el experimento de exploración.

```
mysql> select * from objeto_fisico;
+----------+------------------+-----------------------+-----------------+----------+---------+-----------------+
| nombre   | nombre_conceptual | nombre_caracteristica | nombre_estancia | posicion | id_nodo | numero_aciertos |
+----------+------------------+-----------------------+-----------------+----------+---------+-----------------+
| chair-0  | chair             | NULO                  | estancia-1      |      224 |       0 |               0 |
| chair-1  | chair             | NULO                  | estancia-1      |      225 |       0 |               0 |
| closet-0 | closet            | NULO                  | estancia-1      |      221 |       0 |               0 |
| closet-1 | closet            | NULO                  | estancia-1      |      222 |       0 |               0 |
| closet-2 | closet            | NULO                  | estancia-1      |      223 |       0 |               0 |
| closet-3 | closet            | NULO                  | estancia-1      |      228 |       0 |               0 |
| TABLE-0  | TABLE             | NULO                  | estancia-1      |      226 |       0 |               0 |
| TABLE-1  | TABLE             | NULO                  | estancia-1      |      227 |       0 |               0 |
+----------+------------------+-----------------------+-----------------+----------+---------+-----------------+
8 rows in set (0.00 sec)

mysql> select * from posiciones_objeto_robot;
+-----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
| id  | posicion_obj_x | posicion_obj_y | posicion_obj_z | posicion_rob_x | posicion_rob_y | orientacion_rob_w | orientacion_rob_z |
+-----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
| 221 |          2.33 |         -0.16 |         -1.00 |          0.81 |          0.00 |            0.9999 |            0.0107 |
| 222 |          3.07 |          0.73 |         -1.00 |          1.25 |          0.12 |            0.2000 |            0.9800 |
| 223 |          2.47 |          0.69 |         -1.00 |          1.25 |          0.12 |            0.9463 |            0.3233 |
| 224 |          1.16 |          2.10 |         -1.00 |          1.25 |          0.12 |            0.7281 |            0.6854 |
| 225 |          0.77 |         -0.71 |         -1.00 |          0.19 |          0.68 |           -0.6490 |            0.7610 |
| 226 |          0.33 |         -1.13 |         -1.00 |          0.24 |          0.22 |            0.6721 |           -0.7405 |
| 227 |          0.41 |          0.18 |         -1.00 |          0.98 |         -0.52 |            0.1670 |            0.9860 |
| 228 |         -2.45 |          2.27 |         -1.00 |          0.02 |          0.59 |            0.9020 |            0.4320 |
+-----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
8 rows in set (0.00 sec)
```

Figura 6.32: Prueba 3 de localización de objetos en el mapa

Otra de las ejecuciones es mostrada en la figura 6.32 y de nuevo se añade la representación esquemática de la figura 6.33 con los falsos positivos marcados. Aquí aparece un armario de más y de nuevo la etiqueta mesa identificada en un lugar donde no hay nada. El resto de elementos están situados con un error aceptable si bien es cierto que no aparecen todas las sillas.

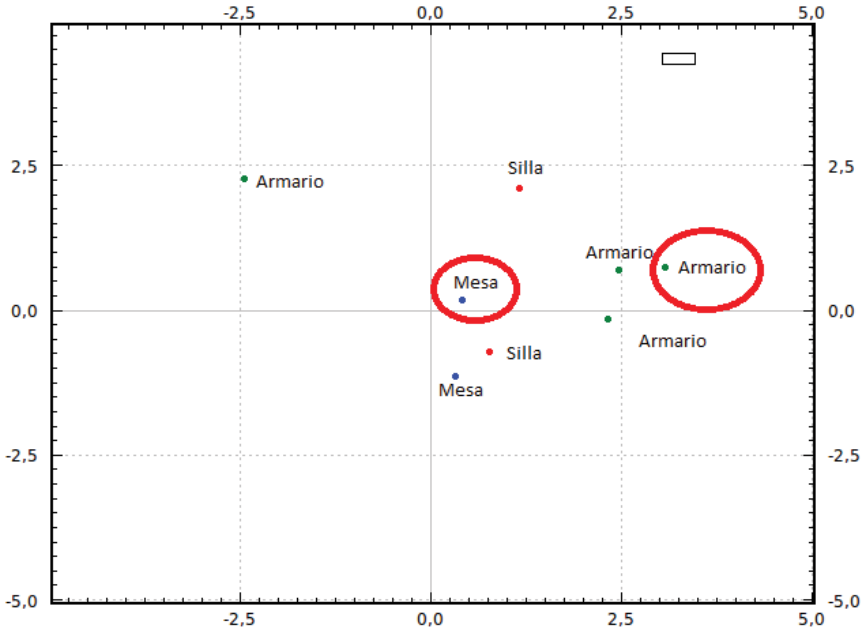


Figura 6.33: Mapa esquemático de los objetos localizados en la prueba 3.

Por último, en la figura 6.34 y su correspondiente mapa esquemático de la figura 6.35 se echa en falta uno de los armarios mientras que aparece otro en una zona incorrecta. Además, aparece identificada una etiqueta de ordenador fuera de los límites de la carpa.

Estos experimentos han mostrado dos problemas cuya solución está contemplada:

- Objetos no detectados. Por la razón que sea (falsos negativos, objeto que no entró en el campo de visión de la cámara o entraron en un ángulo complicado, etc), algunos objetos no fueron detectados en algunos experimentos. Es razonable pensar que con más tiempo de exploración las probabilidades de ser detectados aumentan drásticamente. No se considera un problema.

```
mysql> select * from objeto_fisico;
+-------------+-------------------+-----------------------+-----------------+----------+---------+-----------------+
| nombre      | nombre_conceptual | nombre_caracteristica | nombre_estancia | posicion | id_nodo | numero_aciertos |
+-------------+-------------------+-----------------------+-----------------+----------+---------+-----------------+
| chair-0     | chair             | NULO                  | estancia-1      |        3 |       0 |               0 |
| closet-0    | closet            | NULO                  | estancia-1      |        4 |       0 |               0 |
| closet-1    | closet            | NULO                  | estancia-1      |        5 |       0 |               0 |
| closet-2    | closet            | NULO                  | estancia-1      |        7 |       0 |               0 |
| ORDENADOR-0 | ORDENADOR         | NULO                  | estancia-1      |        6 |       0 |               0 |
| TABLE-0     | TABLE             | NULO                  | estancia-1      |        8 |       0 |               0 |
+-------------+-------------------+-----------------------+-----------------+----------+---------+-----------------+
6 rows in set (0.00 sec)

mysql> select * from posiciones_objeto_robot;
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
| id | posicion_obj_x | posicion_obj_y | posicion_obj_z | posicion_rob_x | posicion_rob_y | orientacion_rob_w | orientacion_rob_z |
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
|  3 |          2.25 |          1.46 |         -1.00 |          0.05 |          0.00 |            0.9975 |            0.0701 |
|  4 |          2.24 |         -0.01 |         -1.00 |          0.34 |          0.04 |            0.9976 |            0.0696 |
|  5 |          2.66 |          0.60 |         -1.00 |          1.00 |          0.22 |            0.9680 |            0.2511 |
|  6 |          4.14 |          3.85 |         -1.00 |          1.00 |          0.22 |            0.2140 |            0.9770 |
|  7 |          1.90 |          1.61 |         -1.00 |          1.00 |          0.23 |            0.4860 |            0.8740 |
|  8 |          0.00 |         -1.01 |         -1.00 |          0.31 |          0.55 |            0.8131 |           -0.5821 |
+----+---------------+---------------+---------------+---------------+---------------+-------------------+-------------------+
6 rows in set (0.00 sec)
```

Figura 6.34: Prueba 4 de localización de objetos en el mapa.

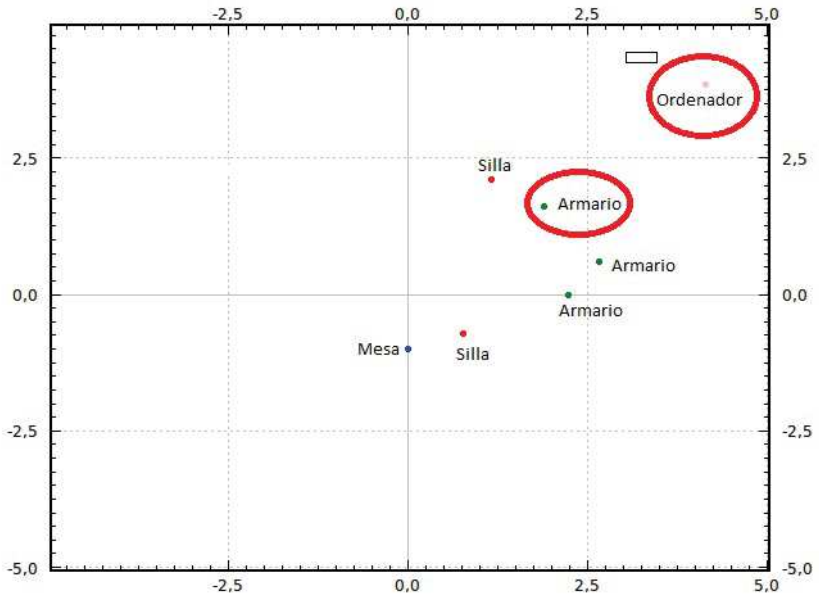


Figura 6.35: Mapa esquemático de los objetos localizados en la prueba 4.

- Falsos positivos. Las medidas tomadas para evitar los falsos positivos (referencia a la sección donde lo explique) están orientadas a evitar fallos puntuales del sistema de percepción, pero no a evitar un fallo persistente. Es decir, que si desde un punto concreto del mapa el sistema detecta un objeto inexistente porque siempre se confunde con un objeto real, el fallo no es detectado. Esto es porque da igual el tiempo que pase y las muestras que se tomen estando estáticos en ese lugar: el sistema siempre cree que ahí hay un objeto que no existe. La solución es comprobar la existencia del objeto desde otro ángulo, aunque eso tiene el riesgo de que un objeto que sí existe no sea percibido desde otro ángulo. Se ha considerado que la solución a este problema puede requerir un trabajo más exhaustivo y se ha iniciado el desarrollo de una funcionalidad que compruebe la detección de un objeto desde varios ángulos, pero no será incluido en esta tesis.

6.4.2. Clasificación de la estancia

Esta prueba se ha realizado a continuación de la prueba de ubicación de objetos en el mapa. Por lo tanto se parte de un entorno explorado con sillas, mesas y armarios. El objetivo es introducir un objeto que clasifique la estancia. Se ha elegido introducir el objeto *ordenador* y su sistema de detección elegido ha sido emplear una etiqueta de ARToolkit. El sistema tiene la información de que los ordenadores se encuentran en los despachos y que el lugar en el que se encuentra es la estancia-1.

La ejecución del experimento se lleva a cabo en dos pasos. Primero se presenta al robot una caja con la etiqueta asociada previamente al objeto *ordenador* (Figura 6.36). Este objeto es detectado por el sistema y tenido en cuenta por el módulo explorador. En la figura 6.37 se muestra la salida por pantalla del módulo explorador cuando está percibiendo el ordenador.

El segundo paso es comprobar que en la base de datos del robot ahora aparece que la estancia en la que se encontraba, la estancia-1, está relacionada con *despacho*. En la tarea de la clasificación de estancias, el sistema emplea la información almacenada en la tabla *estancia_objeto_conceptual*. En la figura 6.38 se muestra una captura de pantalla de las consultas en mysql que se realizaron tras la exposición de la etiqueta *ordenador* al robot. Primero se muestra el contenido de la tabla mencionada, donde

Figura 6.36: Se introduce en el campo visual del robot una caja con una etiqueta que corresponde al objeto *ordenador*. El robot interpreta que está percibiendo un ordenador.

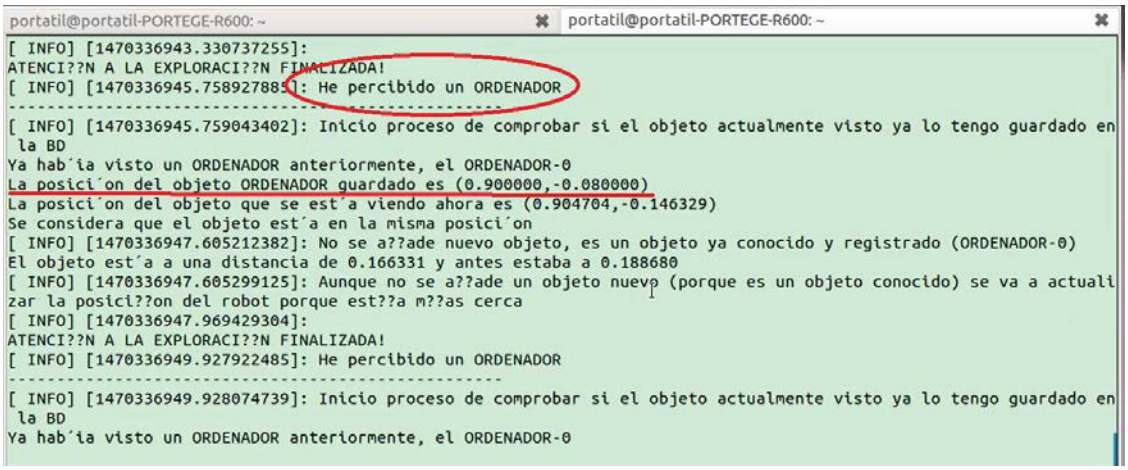


Figura 6.37: Mensajes en pantalla del módulo explorador.

se relacionan una serie de objetos con un conjunto de tipos de estancia. En este experimento se ha considerado un entorno con los tipos de estancia AULA, CAFETERÍA y DESPACHO. El ordenador sólo se encuentra en las estancias de tipo DESPACHO. La flecha roja de la figura señala que el sistema incluyó en su base de datos la relación de la estancia-1 con DESPACHO después de que la etiqueta correspondiente a un ordenador fuera detectada en dicha estancia.

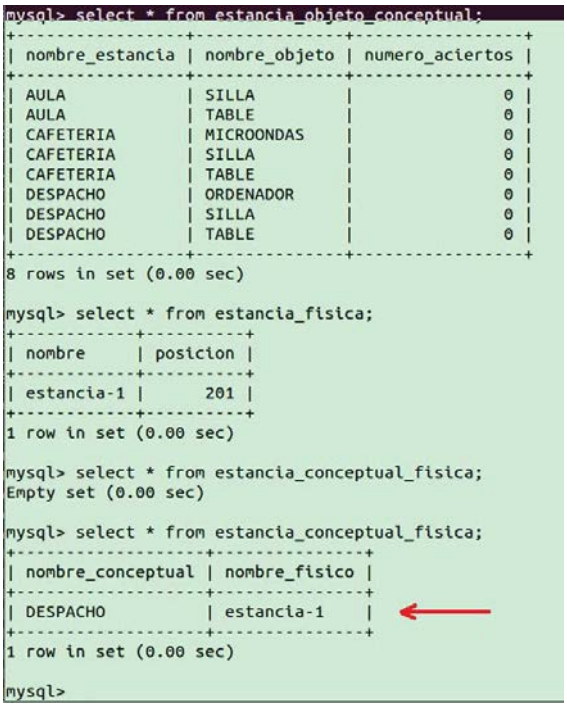


Figura 6.38: Captura de pantalla de consultas en mysql en el experimento de clasificación de estancias.

CAPÍTULO 7

Conclusiones

Al finalizar el desarrollo de esta tesis doctoral llega el momento de analizar tanto los objetivos que se han alcanzado y los aportes que quedan constatados como lo que ha quedado encauzado para completar en un futuro como líneas de investigación abiertas.

7.1. Conclusiones

El objetivo principal propuesto en el comienzo de esta tesis ha sido alcanzado teniendo en cuenta los resultados experimentales del capítulo 6. Se ha obtenido un sistema de navegación capaz de interaccionar con el usuario con interfaces de distinta naturaleza (voz o teclado), al que se le ha solicitado ir a distintos destinos para lo que se ha empleado un sistema de razonamiento que ha conseguido elegir el destino de bajo nivel más adecuado, siendo este topológico o geométrico. El sistema ha respondido correctamente ante peticiones de objetos, de acciones, de valoraciones subjetivas (como ir a un lugar divertido) o de estancias genéricas. Para ello, el sistema ha sido capaz de entender las relaciones entre todas las entidades del entorno, aprender conceptos nuevos de distintas maneras, entre las que han estado las preguntas al usuario, las

consultas a servicios de web semánticas y las clasificaciones de estancias de manera autónoma por el robot. Estas clasificaciones se han realizado en función de objetos detectados y de reconocimiento de escenas, que ha sido incluido en la habilidad de exploración del robot.

Además, trabajando en el camino hacia la consecución de los objetivos propuestos inicialmente, se han alcanzado una serie de contribuciones que van en varias direcciones:

- *Mejoras de modelos ontológicos.* El modelo ontológico presentado en esta tesis describe con suficiente precisión entornos creados por humanos para que un robot pueda navegar por ellos. La representación de la ontología se ha mostrado en un diagrama entidad-relación donde se combinan entidades físicas y conceptuales. La gestión del conocimiento semántico es flexible y adaptable a distintos entornos. Se contemplan conceptos novedosos como significados subjetivos sobre las acciones que se realizan con ciertos objetos, y las relaciones entre las utilidades de los objetos y la estancia donde se encuentran. Se considera también que las características de los objetos pueden cambiar su función y, por tanto, su localización. Además, el modelo es ampliable con cualquier tipo de relación entre objetos.

- *Mejoras de prestaciones en sistemas de razonamiento.* Este trabajo ha demostrado que con una herramienta sencilla como lo es una base de datos relacional, se puede implementar la ontología necesaria para llevar a cabo una navegación semántica. Además esto aporta que personas sin conocimientos en lenguajes formales y formulación de reglas, puedan gestionar las relaciones y conceptos del entorno. Esto facilita a un usuario no experto la capacidad de definir por sí mismo los datos del entorno. El sistema de razonamiento propuesto también ha resultado ser más rápido que las alternativas presentadas con motores de razonamiento que trabajan con Prolog, puesto que se aprovecha la velocidad de las bases de datos.

- *Inclusión de nuevos enfoques para la clasificación de lugares.* Este trabajo ha significado un inicio en el camino sin explotar sobre las posibilidades de inferir la utilidad de un lugar en función del comportamiento de la personas que están

en él. Otros sistemas se basan únicamente en las características físicas del lugar, añadiendo información como los objetos que se encuentran en una ubicación o detalles sobre la estancia en sí. El enfoque de catalogar un lugar centrando la atención en lo que hacen las personas que están allí es algo novedoso que amplía la potencia de los clasificadores de estancias.

- *Integración de sistemas.* El navegador desarrollado en esta tesis se crea a partir de un diseño modular que permite integrar distintos componentes con facilidad. Esto permite emplear el razonador que se desee, así como cualquier navegador de bajo nivel hecho en ROS, otros sistemas de interacción humano-robot y distintos dispositivos de información sensorial. La integración perpectiva es especialmente interesante por presentar la capacidad de agregar múltiples sistemas de detección de objetos.

7.2. Trabajos futuros

La línea de investigación seguida en esta tesis se considera fructífera debido a las posibilidades de investigación que quedan abiertas. La construcción del navegador semántico desarrollado ha generado un considerable volumen de trabajo y se consideran algunos objetivos futuros que podrían mejorar o ampliar el sistema propuesto. Los principales son:

- Actualizaciones del mapa simultáneas con la navegación. Sería deseable que el explorador pudiera actualizar el mapa según se mueve por el entorno. Esto además solucionaría problemas con falsos positivos de objetos en el entorno, puesto que el sistema comprobaría que estuvieran en el entorno los objetos reconocidos previamente.

- Ordenación configurable de objetivos semánticos. El sistema actual contempla la ordenación de objetivos semánticos en función del número de aciertos. Es decir, que si el planificador encuentra varios objetos o lugares distintos que encajan con la petición del usuario, los ordena en función de las veces anteriores que cada objetivo fue seleccionado. Sin embargo sería interesante poder ordenar por otros criterios, como la distancia a cada uno de los objetivos.

- Empleo de la posición semántica en la tarea de reconocimiento de objetos. Los sistemas de detección de objetos empleados podrían ver sus porcentajes de éxito aumentados si se incluye la información semántica del lugar en el que se encuentran. Por ejemplo, un televisor y un microondas podrían ser difícilmente distinguibles entre sí si se emplea una detección de contornos, al ser dos objetos con similares proporciones. Pero incluyendo la información del lugar en el que se encuentran cada uno de ellos (cocina o salón), la detección de objetos podría ser más eficiente.

- Mejora del sistema de detección de personas. En el sistema de etiquetado de lugares basado en lo que hacen las personas en el entorno, el detector de personas utilizado se basa en una detección de piernas, lo que dificulta el reconocimiento de personas sentadas. Como trabajo futuro queda incluir detección de caras además de un reconocedor de expresiones faciales para tener más información útil para decidir qué acciones están realizando las personas.

- Mejora del aprovechamiento semántico en el bajo nivel de navegación. La integración de los conceptos semánticos en los navegadores geométricos y topológicos se han mostrado como posibles lineas de investigación futuras, en las que ya se han dado continuidad a través de nuevas tesis doctorales.

- Reconocimiento de acciones o utilidades de los objetos. Se ha estudiado la posibilidad de incluir sistemas clasificadores que puedan relacionar un objeto con utilidades en función de su forma, de la interacción con una persona y de otras características. Esto aportaría más información al robot.

- Profundizar en la inclusión de predicciones de destino. Se ha presentado una primera aproximación de un sistema enfocado a predecir el destino posible de un usuario, empleando árboles de decisión. Un trabajo futuro llevaría a investigar en esta línea y ofrecer un sistema más elaborado y probado.

- Inclusión de incertidumbre. Contemplar la información con cierto grado de incertidumbre puede enriquecer el sistema. Sería interesante desarrollar algunos experimentos con este concepto.

En la actualidad hay muchas de estas líneas de investigación ya iniciadas, que están dando lugar a nuevas tesis doctorales que continúan y avalan los trabajos realizados en esta tesis.

At the end of this doctoral thesis we analyze the objectives that have been achieved, the contributions that have been verified, which has been channeled to complete in the future and the research topics open.

7.3. Conclusions

The main objective proposed at the beginning of this thesis has been reached taking into account the experimental results of chapter 6. A navigation system capable of interacting with the user with interfaces of different nature (voice or keyboard) has been obtained. This system has been requested to go to different destinations. To this end, a reasoning system has been used, which has succeeded in choosing the most appropriate low-level destination. The low-level navigation systems used have been topological and geometric. The system has responded correctly to requests for objects, actions, subjective assessments (such as going to a fun place) or generic rooms. To do this, the system has been able to understand the relationships between all entities in the environment and learn new concepts in different ways. The learning ways have included questions to the user, queries to semantic web services and classifications of rooms autonomously by the robot. These classifications have been made based on

detected objects and scene recognition. This has been included in the robot's exploring ability.

In addition, the work towards the achievement of the objectives initially proposed, has allowed to provide several secondary contributions:

- *Ontological models improvements.* The ontological model presented in this thesis describes with enought precision human-created environments so that a robot can navigate them. The representation of the ontology has been shown in an entity-relationship diagram where physical and conceptual entities are combined. The management of semantic knowledge is flexible and adaptable to different environments. Novel concepts are considered as meanings about the actions that are performed with certain objects, and the relations between the utilities of the objects and the romm where they are contained. It is also considered that the characteristics of objects can change their function and, therefore, their location. In addition, the model is expandable with any type of object relationship.

- *Performance Improvements in Reasoning Systems.* This work has demonstrated that with a relational database can be implemented the ontology necessary to perform a semantic navigation. In addition this contributes that people without knowledge in formal languages and formulation of rules, can manage the relations and concepts of the environment. This makes it easier for a non-expert user to define the environment data by himself. The proposed reasoning system has also proved to be faster than the alternatives presented with reasoning engines that work with Prolog, since it exploits the speed of the databases.

- *Inclusion of new approaches to the place classification.* This work is a start in an untapped approach to the possibilities of inferring the usefulness of a place depending on the behavior of the people who are in it. Other systems are based solely on the physical characteristics of the place, adding information such as objects in a location or details about the room. The approach of labeling a place by focusing attention on what people are doing is something new that extends the power of the room classifiers.

- *System integration.* The navigation system developed in this thesis is created from a modular design that allows integrating different components. This allows the use of any reasoning system, as well as any low-level navigation system made in ROS, other human-robot interaction systems and different sensory information devices and technologies. Perceptual integration is especially interesting because it has the ability to add multiple object detection systems.

7.4. Future work

The research followed in this thesis is considered fruitful due to the work that has produced and those that have remained pending. The construction of the developed semantic navigation system has generated a great volume of work. Some future objectives are commented on because they could improve or expand the proposed system. The main ones are:

- Simultaneous semantic map updates while moving. It would be desirable for the navigation system to update the map as the robot moves around the environment. This would also solve problems with false positives of object recognition in the environment, since the system would verify that previously recognized objects were in the environment.

- Sort the semantic objectives in a configurable way. The current system contemplates semantic objectives sorted according to the number of hits. If the planner encounters several objects or different places that match with the user's request, the system sort them based on the previous times that each target was selected. However it would be interesting to be able to sort by other criteria, such as the distance to each of the objectives.

- Use the semantic position in the object recognition task. Object detection systems used could increase their success rates if the semantic information of the place in which they are found is included. For example, a TV and a microwave could be difficult to distinguish from each other if a contour detection is used, because two objects with similar shape. But including the information of the

place where each of them (kitchen or living room), the detection of objects could be more efficient.

- Improve the detection system for people. In the place labeling system based on what people do, the people detector used is based on a detection of legs. This makes it difficult to recognize sitting people. Future work includes face detection plus a facial expression recognizer to have more useful information to decide what actions people are doing.

- Improve the semantic use of the low level of navigation. The integration of semantic concepts in geometric and topological browsers has been shown as possible future research fields.

- Recognize actions or utilities of objects. The possibility of including classifying systems that can relate an object to utilities depending on their form, interaction with a person and other characteristics is a future field of work. This would bring more information to the robot.

- Deepen the target predictions. A first approximation of a system focused on predicting the possible destination of a user has been presented. Decision trees have been used. Researching in this field and offering a more elaborate and powerful system is a future work.

- Include uncertainty. Contemplating information with some degree of uncertainty can improve the system. It would be interesting to develop some improvements with this concept.

At present many investigations in these fields have already begun. These are giving rise to new doctoral theses, which continue and endorse the work done in this thesis.

Bibliografía

[1] Eugenio Aguirre, Miguel Garcia-Silvente, y Javier Plata. Leg Detection and Tracking for a Mobile Robot and Based on a Laser Device, Supervised Learning and Particle Filtering. En Manuel A. Armada, Alberto Sanfeliu, y Manuel Ferre, eds., *ROBOT2013: First Iberian Robotics Conference*, tomo 252, págs. 433–440. Springer International Publishing, Cham, 2014. ISBN 978-3-319-03412-6 978-3-319-03413-3.

[2] B. Alexander, K. Hsiao, C. Jenkins, B. Suay, y R. Toris. Robot web tools (ros topics). *Robotics and Automation Magazine*, 2012.

[3] Philipp Althaus y Henrik I. Christensen. Behaviour coordination for navigation in office environments. En *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, tomo 3, págs. 2298–2304. IEEE, 2002.

[4] Kai O. Arras, Óscar Martínez Mozos, y Wolfram Burgard. Using boosted features for the detection of people in 2d range data. En *Robotics and Automation, 2007 IEEE International Conference on*, págs. 3402–3407. IEEE, 2007.

[5] K.O. Arras. *Featur-based robot navigation in know and unknow environments.* Tesis Doctoral, Swiss Federal Institute of Technology Lausanne (EPFL), 2003.

[6] C. Astua, R.Barber, J. Crespo, y A. Jardon. Object detection techniques applied on mobile robot semantic navigation. *Sensors*, 14(4):6734 – 6757, 2014.

[7] Carlos J. Astúa Elizondo. *Object Detection For Mobile Robots*. Proyecto Fin de Carrera, 2013.

[8] A. Aydemir, M. Göbelbecker, A. P. K. S.belbecker, A. Pronobis, K. Sjöö, y P. Jensfelt. Plan-based object search and exploration using semantic spatial knowledge in the real world. En *Proceedings of the 5th European Conference on Mobile Robots (ECMR11)*. 2011.

[9] R. Barber. *Desarrollo de una arquitectura para robots móviles autónomos. Aplicación a un sistema de navegación topológica*. Tesis Doctoral, 2000.

[10] Herbert Bay, Andreas Ess, Tinne Tuytelaars, y Luc Van Gool. *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[11] Patrick Beeson, Matt MacMahon, Joseph Modayil, Aniket Murarka, Benjamin Kuipers, y Brian Stankiewicz. Integrating Multiple Representations of Spatial Knowledge for Mapping, Navigation, and Communication. En *Interaction Challenges for Intelligent Assistants*, págs. 1–9. 2007.

[12] Michael Beetz, Daniel Beßler, Jan Winkler, Jan-Hendrik Worch, Ferenc Balint-Benczedi, Georg Bartels, Aude Billard, Asil Kaan Bozcuoglu, Zhou Fang, Nadia Figueroa, Andrei Haidu, Hagen Langer, Alexis Maldonado, Ana Lucia Pais Ureche, Moritz Tenorth, y Thiemo Wiedemeye. Open robotics research using web-based knowledge services. En *IEEE International Conference on Robotics and Automation (ICRA)*. 2016.

[13] J.L. Blanco, J. González, y J.-A. Fernández-Madrigal. Subjective local maps for hybrid metric-topological slam. *Robotics and Autonomous Systems*, 57:64–74, 2009.

[14] Clara Gómez Blázquez. *Topological Navigation System Applied to a Developed Robot*. Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2016.

[15] Nico Blodow, Lucian Cosmin Goron, y Zoltan-Csaba Marton. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2011.

[16] V.V. Borisov, V.V. Kruglov, y A.C. Fedulov. Fuzzy models and networks. En *Publishing House Telekom*. 2004.

[17] W.N. Borst. *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. Tesis Doctoral, 1997.

[18] Michael S. Brandstein, John E. Adcock, y Harvey F. Silverman. A closed-form location estimator for use with room environment microphone arrays. *Speech and Audio Processing, IEEE Transactions on*, 5(1):45–50, 1997.

[19] R. Brown. How shall a thing be called? *Psychological Review*, 1958.

[20] E. Brunskill, T. Kollar, y N. Roy. Topological mapping using spectral clustering and classification. En *IEEE/RSJ Conference on Robots and Systems*. 2007.

[21] Gail A. Carpenter, Stephen Grossberg, y David B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4:759–771, 1991.

[22] G. Cestnik, I. Konenenko, y I. Bratko. A knowledge elicitation tool for sophisticated users. En *Progress in Machine Learning*. 1987.

[23] Pi Sheng CHANG. Acoustic source location using a microphone array. 2002.

[24] R. Chatila y J.P. Laumond. Position referencing and consistent world modeling for mobile robots. En *IEEE International Conference on Robotics and Automation*. 1985.

[25] A. Chong y K.W.Wong. On the fuzzy cognitive map attractor distance. En *IEEE Congress on Evolutionary Computation 1*. 2007.

[26] H. Choset y K. Nagatani. Topological simultaneous localization and mapping (slam): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17:125–137, 2001.

[27] Jonas Cleveland, Dinesh Thakur, Philip Dames, Cody Phillips, Terry Kientz, Kostas Daniilidis, John Bergstrom, y Vijay Kumar. An automated system for semantic object labeling with soft object recognition and dynamic programming segmentation. En *Automation Science and Engineering (CASE), 2015 IEEE International Conference on*, págs. 683–690. IEEE, 2015.

[28] S. Coradeschi y A. Saffiotti. An introduction to the anchoring problem. *Robotics and Autonomous Systems (RAS)*, 43(2-3):85–96, 2003. Special issue on perceptual anchoring. Online at http://www.aass.oru.se/Agora/RAS02/.

[29] Jonathan Courbon, Youcef Mezouar, y Philippe Martinet. Indoor navigation of a non-holonomic mobile robot using a visual memory. *Autonomous Robots*, 2008.

[30] Jonathan Crespo, Ramón Barber, y O. Martinez Mozos. An Inferring Semantic System Based on Relational Models for Mobile Robotics. En *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, págs. 83–88. IEEE, 2015. ISBN 978-1-4673-6991-6.

[31] Jonathan Crespo, Ramón Barber, y O.M. Mozos. Relational model for robotic semantic navigation in indoor environments. *Journal of Intelligent and Robotic Systems*, págs. 1–23, 2017.

[32] Jonathan Crespo, Clara Gómez, Alejandra Hernández, y Ramón Barber. A semantic labeling of the environment based on what people do. *Sensors*, 17(2), 2017.

[33] G. Csurka, C. Dance, L. Fan, J. Willamowski, y C. Bray. Visual categorization with bags of keypoints.in. En *Proceedings of the Workshop on Statistical Learning in Computer Vision (ECCV),*. 2004.

[34] M. Cummins y P. M. Newman. Highly scalable apearance-only slam- fab-map 2.0. En *Proc of RSS*. 2009.

[35] F. Dayoub, G. Cielniak, y T. Duckett. A sparse hybrid map for vision-guided mobile robots. En *Proc. of ECMR'11*. 2011.

[36] Miriam Lopez de-la Calleja, Takayuki Nagai, Muhammad Attamimi, Mariko Nakano-Miyatake, y Hector Perez-Meana. Object detection using surf and superpixels. *Journal of Software Engineering and Applications*, 2013.

[37] Romain Drouilly, Patrick Rives, y Benoit Morisset. Semantic Representation For Navigation In Large-Scale Environments. En *IEEE Int. Conf. on Robotics and Automation, ICRA'15*. 2015.

[38] R. O. Duda, P. E. Hart, y D. G. Stork. Pattern classification. *Wiley Interscience*, 2000.

[39] V. Egido. *Sistema de navegación topológica para robots móviles autónomos*. Tesis Doctoral, 2003.

[40] S. Ekvall y D.Kragic. Receptive field coocurrence histograms for object detection. En *IEEE/RSJ International Conference Intelligent Robots and Systems (IROS)*. 2005.

[41] S. Ekvall, D. Kragic, y P. Jensfelt. Object detection and mapping for service robot task, robotica. *International Journal of information, Education and Research in Robotics and Artificial Intelligence*, 25:175–187, 2007.

[42] S. Friedman, H. Pasula, y D. Fox. Voronoi random fields: extracting the topological structure of indoor environments via place labeling. En *International Joint Conference on Artificial Intelligence*. 2007.

[43] C. Galindo, J-A. Fernández-Madrigal, J. González, y A. Saffiotti. Robot task planning using semantic maps. En *Robotics and Autonomous Sistems*. 2008.

[44] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. A. Fernández-Madrigal, y J.González. Multi-hierarchical semantic maps for mobile robotics. En *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005.

[45] Cipriano Galindo y Alessandro Saffiotti. Inferring robot goals from violations of semantic knowledge. *Robotics and Autonomous Systems*, 61(10):1131 – 1143, 2013.

[46] Thomas D. Garvey. *Perceptual strategies for purposive vision*. 1976.

[47] Guglielmo Gemignani, Roberto Capobianco, Emanuele Bastianelli, Domenico Daniele Bloisi, Luca Iocchi, y Daniele Nardi. Living with robots: Interactive environmental knowledge acquisition. *Robotics and Autonomous Systems*, 2016.

[48] G.Grisetti, C.Strachniss, y W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. En *IEEE Transactions on Robotics*. 2007.

[49] Nils Goerke y Sven Braun. Building semantic annotated maps by mobile robots. En *Towards Autonomous Robotic Systems*. 2009.

[50] Ken Goldberg y Ben Kehoe. Cloud robotics and automation: A survey of related work. Inf. téc., EECS Department, University of California, Berkeley, 2013.

[51] Asunción Gómez-Pérez, Mariano Fernández-López, y Oscar Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. (Advanced Information and Knowledge Processing)*. Springer-Verlag New York, Inc., 2007. ISBN 1846283965.

[52] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human and Computer Studies*, 43:907–928, 1995.

[53] N. Guarino. Formal ontology and information systems. En *Proceedings of the First International Conference On Formal Ontology In Information Systems*. 1998.

[54] Sachithra Hemachandra, Thomas Kollar, Nicholas Roy, y Seth J. Teller. Following and interpreting narrated guided tours. En *ICRA*. 2011.

[55] A. C. Hernández, C. Gómez, J. Crespo, y R. Barber. Object detection applied to indoor environments for mobile robot navigation. *Sensors*, págs. 1 – 26, 2016.

[56] Alejandra Hernández, Clara Gómez, Jonathan Crespo, y Ramón Barber. A home made robotic platform based on theo jansen mechanism for teaching robotics. En *Proceedings of the 10th Annual International Technology, Education and Development Conference (INTED 2016)*. 2016.

[57] José-Juan Hernández-López, Ana-Linnet Quintanilla-Olvera, José-Luis López-Ramírez, Francisco-Javier Rangel-Butanda, Mario-Alberto Ibarra-Manzanoa, y Dora-Luz Almanza-Ojeda. Detecting objects using color and depth segmentation with kinect sensor. En *Procedia Technology*. 2012.

[58] Alejandra C. Hernández Silva. *Object Detection Applied To Indoor Environments For Mobile Robot Navigation*. Proyecto Fin de Carrera, 2016.

[59] Jonathan Crespo Herrero. *Diseño de un sistema de navegación semántica para robots móviles. Empleo de base de datos relacional como motor de razonamiento.* Proyecto Fin de Carrera, 2012.

[60] Y. Hua, S. Zander, M. Bordignon, y B. Hein. From automationml to ros: A model-driven approach for software engineering of industrial robotics using ontological reasoning. En *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, págs. 1–8. 2016.

[61] A. Jastriebow y K. Poczeta. Analysis of multi-step algorithms for cognitive maps learning. *Bulletin of the Polish Academy of Sciences Technical Sciences*, págs. 133–144, 2014.

[62] Dominik Joho y Wolfram Burgard. Searching for objects: Combining multiple cues to object locations using a maximum entropy model. En *IEEE International Conference on Robotics and Automation (ICRA)*. 2010.

[63] Y. Jung, J.Y. Lee, Y. Kim, J. Park, S.H. Myaeng, y H.C. Rim. Building a large-scale commonsense knowledge base by converting an existing one in a different language. *Computational Linguistics and Intelligent Text Processing*, págs. 23–34, 2007.

[64] N.Y. Khan, B. McCane, y G. Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. En *Proceedings of*

the 2011 International Conference on Digital Image Computing Techniques and Applications (DICTA). 2011.

[65] Dong Wook Ko, Chuho Yi, y Il Hong Suh. Semantic mapping and navigation: A bayesian approach. En *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013.

[66] Thomas Kollar y Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. En *IEEE International Conference on Robotics and Automation (ICRA)*. 2009.

[67] Ioannis Kostavelis, Konstantinos Charalampous, Antonios Gasteratos, y John K. Tsotsos. Robot navigation via spatial and temporal coherent semantic maps. *Engineering Applications of Artificial Intelligence*, 48:173–187, 2016.

[68] Ioannis Kostavelis y Antonios Gasteratos. Learning spatially semantic representations for cognitive robot navigation. *Robotics and Autonomous Systems*, 2013.

[69] Scott Krig. *Computer Vision Metrics: Survey, Taxonomy, and Analysis*. Apress, 2014.

[70] Geert-Jan M. Kruijff, Hendrik Zender, Patric Jensfelt, y Henrik I. Christensen. Clarification dialogues in human-augmented mapping. En *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-robot Interaction*, HRI '06. 2006.

[71] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[72] B.J. Kuipers. The cognitive map: could it have been any other way? En *Spatial Orientation: Theory, Research, and Application*. 1983.

[73] Lars Kunze, Tobias Roehm, y Michael Beetz. Towards semantic robot description languages. En *IEEE International Conference on Robotics and Automation (ICRA)*, págs. 5589–5595. 2011.

[74] M. Laclavk, Z. Balogh, M. Babk, y L HluchY. Agentowl: Semantic knowledge model and agent architecture. *Computing and Informatics*, 25:421–439, 2006.

[75] H. Liu y P. Singh. Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22(4):211–226, 2004.

[76] D.G. López. *Combining object recognition and metric mapping for spatial modeling with mobile robots*. Proyecto Fin de Carrera, Royal Institute of Technology, 2007.

[77] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[78] David V. Lu y William D. Smart. Towards more efficient navigation for robots and humans. En *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference On*, págs. 1707–1713. IEEE, 2013.

[79] Jun Luo, Jinqiao Wang, Huazhong Xu, y Hanqing Lu. Real-time people counting for indoor scenes. *Signal Processing*, 2016.

[80] T. MacNamara. *Mental representations of spatial relations, Cognitive Psychology*. 1986.

[81] David Meger, Per-Erik Forssén, Kevin Lai, Scott Helmer, Sancho McCann, Tristram Southey, Matthew Baumann, James J. Little, y David G. Lowe. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems*, 56(6):503 – 511, 2008.

[82] M. Milford y G. Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24:1038–1153, 2008.

[83] Hans Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 1988.

[84] Enrique Moure. *Dialog System Interface for Semantic Navigation*. Proyecto Fin de Carrera, 2015.

[85] O. M. Mozos, Z.-C. Marton, y M. Beetz. Furniture models learned from the www – using web catalogs to locate and categorize unknown furniture pieces in 3d laser scans. *IEEE Robotics & Automation Magazine*, 18(2):22–32, 2011.

[86] Oscar Martinez Mozos, Cyrill Stachniss, y Wolfram Burgard. Supervised learning of places from range data using adaboost. En *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, págs. 1730–1735. IEEE, 2005.

[87] Oscar Martinez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, y Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[88] M. Muja y D. Lowe. *FLANN - Fast Library for Approximate Nearest Neighbors User Manual*. 2009.

[89] R. Nesaratnam, C. Bala Murugan, y others. Identifying leaf in a natural image using morphological characters. En *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*, págs. 1–5. IEEE, 2015.

[90] Carlos Nieto-Granda, John G. Rogers III, Alexander J. B. Trevor, y Henrik I. Christensen. Semantic map partitioning in indoor environments using regional analysis. En *IEEEl/RSJ International Conference on Intelligent Robots and Systems*. 2010.

[91] Natalya F Noy, Monica Crubézy, Ray W Fergerson, Holger Knublauch, Samson W Tu, Jennifer Vendetti, Mark A Musen, et al. Protege-2000: an open-source ontology-development and knowledge-acquisition environment. En *AMIA Annu Symp Proc*. 2003.

[92] Andreas Nüchter y Joachim Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915 – 926, 2008.

[93] Sageev Oore, Geoffrey E. Hinton, y Gregory Dudek. A Mobile Robot That Learns Its Place. *Neural Computation*, 9(3):683–699, 1997.

[94] S. Patnaik. *Robot Cognition and Navigation: An Experiment with Mobile Robots (Cognitive Technologies)*. 2005.

[95] Liam Paull, Gaetan Severac, Guilherme V. Raffo, Julian Mauricio Angel, Harold Boley, Phillip J. Durst, Wendell Gray, Maki Habib, Bac Xuan Nguyen, S. Veera Ragavan, et al. Towards an ontology for autonomous robots. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[96] P.Beeson, N.K. Jong, y B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. En *IEEE International Conference on Robotics and Automation*. 2005.

[97] J. Peltason, F. H. K. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide, y E. A. Topp. Mixed-initiative in human augmented mapping. En *2009 IEEE International Conference on Robotics and Automation*. 2009.

[98] Maria-Salome Perez y Enrique V. Carrera. Acoustic event localization on an Arduino-based wireless sensor network. En *Communications (LATINCOM), 2014 IEEE Latin-America Conference on*, págs. 1–6. IEEE, 2014.

[99] Rodrigo Polastro, Fabiano Corrêa, Fabio Cozman, y Jun Okamoto Jr. Semantic mapping with a probabilistic description logic. En *Advances in Artificial Intelligence - SBIA 2010*, págs. 62–71. Springer, 2010.

[100] Edson Prestes, Joel Luis Carbonera, Sandro Rama Fiorini, Vitor A. M. Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo Goncalves, Marcos E. Barreto, Maki Habib, Abdelghani Chibani, Sébastien Gérard, Yacine Amirat, y Craig Schlenoff. Towards a core ontology for robotics and automation. *Robot. Auton. Syst.*, 61(11):1193–1204, 2013. ISSN 0921-8890.

[101] A. Pronobis y P. Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. En *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA'12)*. 2012.

[102] A. Pronobis, P. Jensfelt, K. Sjöö, H. Zender, G.-J. M. Kruij, O. M. Mozos, y W. Burgard. *Semantic modelling of space*. 2010.

[103] A. Pronobis, O. Martinez Mozos, B. Caputo, y P. Jensfelt. Multi-modal Semantic Place Classification. *The International Journal of Robotics Research*, 2010.

[104] A. Pronobis, O. M. Mozos, B. Caputo, y P. Jensfelt. Multi-modal semantic place classification. *IJRR*, 29, 2010.

[105] Gonzalez Rafael y Woods Richard. *Digital Image Processing*. 2007.

[106] Gabriele Randelli, Taigo Maria Bonanni, Luca Iocchi, y Daniele Nardi. Knowledge acquisition through human-robot multimodal interaction. *Intelligent Service Robotics*, 2013.

[107] A. Ranganathan. Pliss: Detecting and labeling places using online change-point detection. En *Proc. of RSS'10*. 2010.

[108] K. Rebai, O. Azouaoui, y N. Achour. Bio-inspired visual memory for robot cognitive map building and scene recognition. En *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012.

[109] Karima Rebai, Ouahiba Azouaoui, y Nouara Achour. Bio-inspired visual memory for robot cognitive map building and scene recognition. En *IEEE/RSJ International Conference on Intelligent Robots and Systems*, págs. 2985–2990. IEEE, 2012.

[110] J. Rituerto, A. C. Murillo, y J. Košecka. Label propagation in videos indoors with an incremental non-parametric model update. En *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, págs. 2383–2389. IEEE, 2011.

[111] E. Rosch. *Cognition and Categorization*. Lawrence Erlbaum Associates, 1978.

[112] Axel Rottmann, Oscar Martinez Mozos, Cyrill Stachniss, y Wolfram Burgard. Semantic place classification of indoor environments with mobile robots using boosting. En *in Proc. of the National Conference on Artificial Intelligence (AAAI*. 2005.

[113] Jose-Raul Ruiz-Sarmiento, Cipriano Galindo, y Javier Gonzalez-Jimenez. Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems*, 119:257 – 272, 2017.

[114] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra Kumar Misra, y Hema Swetha Koppula. Robobrain: Large-scale knowledge engine for robots. *CoRR*, abs/1412.0691, 2014.

[115] R. Schapire y Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn*, 1999.

[116] C. Schlenoff, E. Prestes, R. Madhavan, P. Goncalves, H. Li, S. Balakirsky, T. Kramer, y E. Miguelanez. Ieee standard ontologies for robotics and automation. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[117] Drew Schmitt y Nicholas McCoy. Object classification and localization using surf descriptors. *CS 229 Final Projects*, 2011.

[118] Lei Shi, Sarath Kodagoda, y Gamini Dissanayake. Multi-class classification for semantic labeling of places. En *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, págs. 2307–2312. IEEE, 2010.

[119] Wenxia Shi y Jagath Samarabandu. Investigating the performance of corridor and door detection algorithms in different environments. En *Information and Automation, 2006. ICIA 2006. International Conference on*, págs. 206–211. IEEE, 2006.

[120] Vui Ann Shim, Bo Tian, Miaolong Yuan, Huajin Tang, y Haizhou Li. Direction-driven navigation using cognitive map for mobile robots. En *IEEE International Conference on Intelligent Robots and Systems*, págs. 2639–2646. IEEE, 2014.

[121] Christian Siagian y Laurent Itti. Biologically inspired mobile robot vision localization. *IEEE Transactions on Robotics*, 25(4):861–873, 2009.

[122] Insop Song, Federico Guedea, Fakhreddine Karray, Yanqin Dai, y Ibrahim El Khalil. Natural language interface for mobile robot navigation control. En *Intelligent Control, 2004. Proceedings of the 2004 IEEE International Symposium on*, págs. 210–215. IEEE, 2004.

[123] Pedro Sousa, Rui Araujo, y Urbano Nunes. Real-time labeling of places using support vector machines. En *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, págs. 2022–2027. IEEE, 2007.

[124] Guy Steele. *Common LISP: the language*. Elsevier, 1990.

[125] A. Stevens y P. Coupe. *Distorsions in judged spatial relations, Cognitive Psychology*. 1978.

[126] Rainer Stiefelhagen, Christian Fügen, Petra Gieselmann, Hartwig Holzapfel, Kai Nickel, y Alex Waibel. Natural human-robot interaction using speech, head pose and gestures. En *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, tomo 3, págs. 2422–2427. IEEE, 2004.

[127] S. M. Stringer, T. P. Trappenberg, E. T. Rolls, y I. E. T. de Araujo. Self-organizing continuous attractor networks and path integration: One-dimensional models of head direction cells. *Netw.: Comput. Neural Syst.*, 13:217–242, 2002.

[128] Rudi Studer, V.Richard Benjamins, y Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161 – 197, 1998.

[129] Piergiorgio Svaizer, Marco Matassoni, y Mnririzio Omologo. Acoustic source location in a three-dimensional space using crosspower spectrum phase. En *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, tomo 1, págs. 231–234. IEEE, 1997.

[130] S.Vasudevan y R. Siegwart. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. *Robotics and Autonomous Systems (RAS)*, 56:522–537, 2008.

[131] A. Tapus. *Topological SLAM-Simultaneous Localization And Mapping with fingerprints of places*. Tesis Doctoral, Swiss Federal Institute of Technology Lausanne (EPFL), 2005.

[132] Moritz Tenorth y Michael Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *Int. Journal of Robotics Research*, 32(5):566 – 590, 2013.

[133] Moritz Tenorth, Lars Kunze, Dominik Jain, y Michael Beetz. Knowrob-map – knowledge-linked semantic object maps. En *IEEE-RAS International Conference on Humanoid Robots*. Nashville, TN, USA, 2010.

[134] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, págs. 21–71, 1998.

[135] Sebastian Thrun, Wolfram Burgard, y Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[136] B. Tian, V. A. Shim, M. Yuan, C. Srinivasan, H. Tang, y H. Li. Rgb-d based cognitive map building and navigation. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[137] N. Tomatis, I. Nourbakhsh, y R. Siegwart. Hybrid simultaneous localization and map building: A natural integration of topological and metric, robotics and autonomous systems. *Robotics and Autonomous Systems*, 2003.

[138] A. Torralba, K. Murphy, W. Freeman, y M. Rubin. Context-based vision system for place and object recognition. En *Proc. of ICCV'03*. 2003.

[139] I. Ulrich y I. Nourbakhsh. Appearance-based place recognition for topological localization. En *IEEE Int. Conf. on Robotics and Automation, San Francisco*. 2000.

[140] Mike Uschold y Michael Gruninger. Ontologies: principles, methods and applications. *KNOWLEDGE ENGINEERING REVIEW*, 1996.

[141] S. Vasudevan, S. Gächter, V. Nguyen, y R. Siegwart. Cognitive maps for mobile robots, an object based approach. *Robotics and Autonomous Systems*, 55(5):359 – 371, 2007.

[142] S. Vasudevan y R. Siegwart. A bayesian conceptualization of space for mobile robots. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2007.

[143] Shrihari Vasudevan, Stefan Gächter, Viet Nguyen, y Roland Siegwart. Cognitive maps for mobile robots - an object based approach. *Robotics and Autonomous Systems*, 55(5):359 – 371, 2007.

[144] Shrihari Vasudevan, Ahad Harati, y Roland Siegwart. A bayesian conceptualization of space for mobile robots : Using the number of occurrences of objects to infer concepts. En *European Conference on Mobile Robotics (ECMR)*. 2007.

[145] Shrihari Vasudevan y Roland Siegwart. A bayesian approach to conceptualization and place classification: Incorporating spatial relationships (distances) between objects towards inferring concepts. En *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2007.

[146] Julio Vega, Jose María Canas, y Eduardo Perdices. Attentive visual memory for robot navigation. *Journal of Physical agents*, 1(1), 2011.

[147] Pooja Viswanathan, David Meger, Tristram Southey, James J. Little, y Alan K. Mackworth. Automated Spatial-Semantic Modeling with Applications to Place Labeling and Informed Search. En *Canadian Conference on Computer and Robot Vision, 2009. CRV '09.* IEEE, 2009.

[148] Matthew R. Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, y Seth Teller. Learning semantic maps from natural language descriptions. En *Proceedings of the 2013 Robotics: Science and Systems IX Conference*. 2013.

[149] Tingqi Wang y Qijun Chen. Object semantic map representation for indoor mobile robots. En *International Conference on System Science and Engineering (ICSSE)*. 2011.

[150] Jan Wielemaker, Wouter Beek, Michiel Hildebrand, y Jacco van Ossenbruggen. Cliopatria: A logical programming infrastructure for the semantic web. *Semantic Web Journal*, 2015.

[151] Denis F. Wolf y Gaurav S. Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, 2008.

[152] Hao Wu, Guo-hui Tian, Yan Li, Feng-yu Zhou, y Peng Duan. Spatial semantic hybrid map building and application of mobile service robot. *Robotics and Autonomous Systems*, 62(6):923–941, 2014.

[153] J. Wu, H. I. Christensen, y J. M. Rehg. Visual place categorization: problem, dataset, and algorithm. En *Proc. of IROS'09*. 2009.

[154] J. L. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G-J.M. Kruijff, P. Lison, A.Pronobis, K.Sjöö, A. Vrecko, H. Zender, M. Zillich, y D. Skocaj. Self-understanding & self-extension: a systems and representational approach. *IEEE Transactions on Autonomous Mental Development*, 2(4):282–303, 2010.

[155] Shuníchi Yamamoto, Jean-Marc Valin, Kazuhiro Nakadai, Jean Rouat, François Michaud, Tetsuya Ogata, y Hiroshi G. Okuno. Enhanced robot speech recognition based on microphone array source separation and missing feature theory. En *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, págs. 1477–1482. IEEE, 2005.

[156] Stefan Zander, Georg Heppner, Georg Neugschwandtner, Ramez Awad, Marc Essinger, y Nadia Ahmed. A model-driven engineering approach for ROS using ontological semantics. *CoRR*, abs/1601.03998, 2016.

[157] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. M. Kruijff, y W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems (RAS)*, 56:493–502, 2008.