INGENIERÍA INFORMÁTICA

Escuela Técnica Superior de Ingeniería Informática

Curso académico 2007-2008

Proyecto Beca Colaboración M.E.C.

Reconstrucción de escenas en 3D.

Tutor: José María Cañas Plaza

Autor: Julio Manuel Vega Pérez

Agradecimientos

Resumen

Este proyecto describe el diseño e implementación de un robot móvil capaz de realizar una interpretación estructural del entorno usando únicamente información sensorial de tipo visual y odométrico. El comportamiento deseado es la navegación en tiempo real basada en esta interpretación, en lugar de la estrictamente reactiva a los estímulos inmediatos. Optamos por un criterio de diseño fundamentalmente predictivo: el sistema debe anticipar las consecuencias de sus acciones, mostrando una cierta comprensión predictiva de la escena en la que se mueve.

Se proponen soluciones para todos los niveles de la percepción. Con objeto de inferir propiedades euclídeas del espacio, hemos calibrado la cámara propiamente, basándonos en el proyecto desarrollado en este universidad con tal propósito ([Kachach, 2008]). Así, apoyándonos en el formalismo de la geometría proyectiva, aprovechamos también la informacion odométrica del agente autónomo. La percepción de alto nivel, finalmente, se resuelve mediante un ciclo de generación y seguimiento de información, que es mantenida en una representación interna estable y sintonizada con los movimientos del agente.

La principal característica de este mecanismo es la interacción constante entre los procesos perceptivos ascendentes, guiados por los estímulos sensoriales, y los descendentes, guiados por los modelos previos. Todos estos elementos se integran en una arquitectura hardware-software ([Cañas Plaza *et al.*, 2007]) eficiente, modular y flexible, en la que acción y percepción cooperan estrechamente para lograr la robustez y continuidad necesarias en la navegación.

Índice general

Índice de figuras

Capítulo 1
Introducción

Frasecita.
Autor.

1.1. Generalidades

Recientemente ha aumentado el interés en las ciencias de lo artificial, impulsadas, entre otras causas, por el extraordinario avance tecnológico de la informática. En particular, a partir de la segunda mitad del siglo pasado una de las cuestiones que ha recibido más atención es la construcción de robots autónomos capaces de una interacción inteligente con su entorno. Para alcanzar este objetivo, es necesario resolver el problema de la **percepción artificial**. Éste es precisamente el campo de investigación de este proyecto.

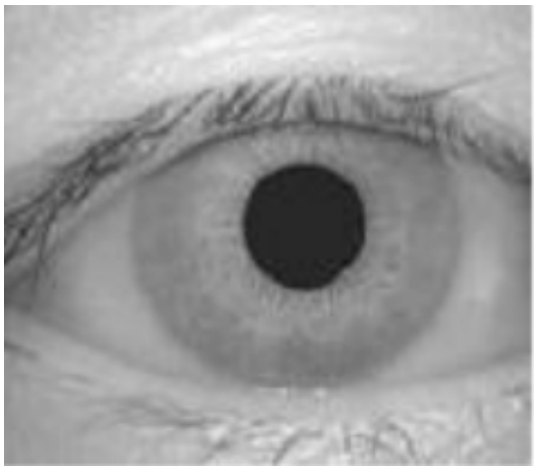

Figura 1.1: Iris humano.

Históricamente, el problema se ha estudiado desde muchos puntos de vista. Una de las primeras hipótesis era que los sistemas cognitivos podían modelarse perfectamente

como sistemas de manipulación simbólica. La percepción se consideraba un simple módulo auxiliar que extraería los símbolos a partir de los estímulos sensoriales, y quedaba por tanto relegada a un segundo plano. Los modelos propuestos utilizaban sistemas de inferencia basados en reglas, manipulaciones formales y métodos de búsqueda heurística. Pero, a menudo, lo que se llamaba *inteligencia artificial* no era más que una búsqueda de isomorfismos sintácticos entre estructuras predeterminadas por el diseñador.

Casi en paralelo surgieron las primeras técnicas de **reconocimiento de patrones** (pattern recognition). Éstas trabajan directamente sobre el espacio de estímulos sensoriales utilizando métodos estadísticos y de optimización). Se trata de detectar automáticamente ciertas propiedades relevantes del mundo externo a través de procesos de aprendizaje computacional, los cuales intentan descubrir regularidades significativas en las variables de entrada a partir de la observación de muchos casos particulares. Algunos modelos estaban inspirados en el funcionamiento del sistema nervioso.

Pronto surgieron los **modelos conexionistas**, capaces de capturar relaciones más complejas mediante la autoorganización de un número elevado de elementos de proceso simples, asíncronos y con interacciones locales. En contraste con los anteriores, todos estos modelos trabajan a un nivel subsimbólico, con representaciones distribuidas. En principio, pueden resolver incluso tareas en las que hay que inferir una cierta estructura composicional en los patrones sensoriales. La mayoría de estas técnicas confían en el principio de inducción ingenuo de que si se resuelven bien muchos casos concretos, se resolverán bien muchos casos futuros.

Más recientemente, la teoría matemática de la generalización ha matizado este principio. Una máquina inductiva probablemente generalizará si resuelve bien muchos más casos concretos que los que sería capaz de *aprender de memoria*. Esta teoría ha permitido construir máquinas extraordinariamente potentes, las SVM, (del inglés **support vector machines**) que controlan automáticamente su capacidad incluso en espacios de propiedades (implícitos) de dimensión arbitrariamente alta. Estas máquinas incluyen como caso particular a otros modelos anteriores y disponen de métodos de aprendizaje más eficientes.

El enfoque puramente inductivo, o empírico, es apropiado para abordar tareas de naturaleza estática, donde es suficiente asociar un espacio de entradas con otro de

salidas, aunque sea a costa de *fuerza bruta muestral*. A través de él se han alcanzado numerosos éxitos en aplicaciones de gran importancia práctica, como el reconocimiento acústico del habla, el reconocimiento óptico de textos escritos, etc. Sin embargo, los tipos de fallo que presentan revelan una frustrante ausencia de comprensión de ciertos aspectos claves del dominio de trabajo. En el caso concreto de los sistemas autónomos, además, este tipo de aprendizaje resulta claramente inviable, ya que es imposible recoger la virtualmente infinita variedad de situaciones que se pueden presentar en la interacción de un robot con un entorno mínimamente complejo.

Intentando superar estas limitaciones se han planteado alternativas que trasladan el énfasis hacia la relación de interdependencia mutua entre el agente y su entorno. Surgen así las aproximaciones basadas en el comportamiento, en las que se conectan los estímulos sensoriales directamente con las acciones sin necesidad de representaciones perceptuales explícitas. Se confía en que la adecuada combinación de diferentes generadores de comportamientos simples produzca un comportamiento global complejo, no explícitamente preprogramado. No se utilizan en ningun momento abstracciones conceptuales, aunque éstas puedan resultar convenientes para describir el comportamiento observado. Se han conseguido resultados prometedores, incluso en robots humanoides, pero parece existir una barrera de complejidad que no es fácil atravesar.

Descendiendo aun más hacia los principios físicos, se ha propuesto también la hipótesis de que los sistemas cognitivos deben considerarse propiamente como sistemas dinámicos. Según estos autores, no debe confundirse la evolución de un sistema natural con una mera simulación computacional de su comportamiento. En lugar de ello, se considera esencial la continuidad de las variables involucradas y su evolución temporal en un espacio de estados, de acuerdo con leyes dinámicas que es razonable expresar mediante ecuaciones diferenciales, en oposición a los algoritmos. Este enfoque tiende a rechazar las representaciones internas, estáticas, y prefiere manejar conceptos de la teoría de sistemas tales como trayectorias del espacio de estados, atractores, bifurcaciones, etc. La aproximación es atractiva, pero llevada al extremo puede limitar en cierto sentido su ámbito de aplicación.

En conclusión, la percepción artificial se ha considerado desde trivial hasta no computable, pasando por los que creen que es innecesario hacerla explícita o simplemente la reducen a un problema estadístico. Como veremos más adelante, hay

incluso quienes señalan la importancia de la percepción de conceptos abstractos y son moderadamente optimistas respecto a la resolución del problema en el futuro. Tales discrepancias nos advierten de que el problema podría estar incluso mal planteado.

1.2. El criterio predictivo

Preguntas como ¿qué es la percepción?, ¿qué produce?, ¿quién la utiliza?, etc., quizá sean prematuras dada la relativa juventud de este campo de investigación. Posiblemente en este momento sea preferible replantear la cuestión de un modo algo más directo y operacional. ¿Qué tipo de procesos pueden contribuir a un comportamiento bien integrado con el entorno? Este punto de vista sugiere inmediatamente una idea que consideramos muy prometedora, a pesar de su simplicidad: *es bueno predecir el futuro, al menos a corto plazo*. Esta afirmación de sentido común es difícilmente rechazable. Nosotros añadiremos el siguiente matiz: la predicción de estados futuros del entorno realizada a partir de los estímulos sensoriales sólo puede contrastarse mediante la propia información sensorial. Esto significa que el intento de predicción explícita de propiedades del entorno podría ser incorrecto. Estas propiedades son infinitas y la selección del diseñador podría sesgar inconscientemente el diseño. Seguramente sea más fructífero tratar de anticipar directamente los estímulos sensoriales. Si esto se consigue a escalas de tiempo suficientemente largas, será difícil negar que se han captado propiedades relevantes del entorno, al menos desde el punto de vista que a nosotros nos interesa, el del propio agente.

En definitiva, dado un dominio de trabajo, y a un nivel determinado por la complejidad del mismo, el criterio descrito puede ayudarnos a juzgar el grado de comprensión predictiva del entorno alcanzada por un robot. En última instancia, por tanto, este criterio puede servir como guía para diseñar sistemas de percepción artificial.

1.3. Representaciones abstractas

Admitamos, pues, que anticipar los estímulos sensoriales es en general deseable para un agente autónomo. El paso siguiente, en nuestra opinión, es entonces casi obligado: necesitamos la mediación de ciertas representaciones abstractas. Hay varios argumentos que sostienen esta afirmación:

- En entornos de complejidad moderada, la evolución de la información sensorial es definitivamente inexplicable sin la ayuda de construcciones conceptuales

completamente ajenas a la dinámica local del espacio de estímulos. Pensemos, por ejemplo, en la aparición y desaparición de manchas de color en una imagen debidas a un objeto que gira en el espacio.

- El uso de abstracciones puede generar predicciones a una velocidad mucho mayor que la que permitiría el tratamiento de la masa sensorial original. De esta forma se reduce significativamente la latencia del ciclo entre la percepción y la acción, aumentándose así la *inmersión* del agente en el entorno. Esto es particularmente relevante cuando los recursos de procesamiento son limitados.

- El contexto espacio-temporal de la información sensorial determina en gran parte la capacidad de predicción. Por ejemplo, dependiendo de aspectos como la iluminación o los objetos vecinos, un objeto de color bien definido puede reflejar luz de prácticamente cualquier otro color. En el extremo opuesto, la dinámica del entorno podría también depender de diferencias arbitrariamente pequeñas en el espacio de estímulos.

Pero, más allá de los aspectos predictivos, hay otras razones de peso que justificarían el uso este tipo de representaciones en los sistemas de percepción artificial. Existen tareas esenciales para un agente autónomo que no parece sencillo resolver sin recurrir a este tipo de abstracciones conceptuales. Un ejemplo es el reconocimiento desde un punto de vista diferente de lugares previamente visitados, posiblemente con pequeños cambios. No es posible establecer una métrica apropiada para esto sobre el espacio de estímulos.

Ya en el ámbito de los sistemas naturales, un argumento adicional sería que muchas ilusiones ópticas no hacen si no evidenciar fallos de interpretación. Podríamos decir que las sensaciones básicas son detecciones de magnitudes elementales en la realidad, y que la percepción consiste en imponer estructura de interpretación al subconjunto de las sensaciones que requieren nuestra atención. Las ilusiones ópticas se basan a menudo en los fallos en la imposición de esta estructura (véase, por ejemplo, la imagen siguiente). Esta dualidad entre información sensorial y organización coherente de los datos tiene también cierta tradición filosofica. Kant, por ejemplo, dividía la percepción en sensibilidad y entendimiento. Parafraseando uno de sus escritos: «*conceptos sin perceptos están vacíos; perceptos sin conceptos son ciegos*».

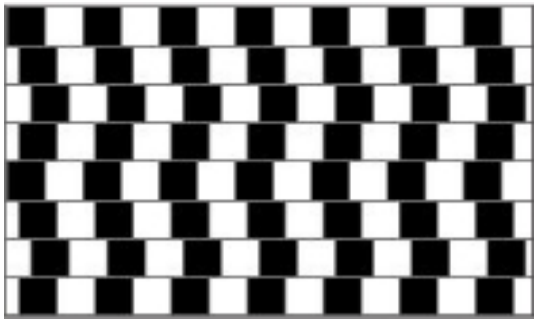

Figura 1.2: Ejemplo de ilusión óptica.

Las representaciones, sin embargo, no pueden ser copias detalladas de la realidad. Caeríamos en una regresión infinita en la que aún tendría que percibirse lo importante de esa representación. La sensación de detalle efectiva sólo se debe a que estamos continuamente informados de cualquier cambio, y se reconstruye inmediatamente la interpretación de la masa sensorial que cae bajo el área de interés. Ver no es crear una representación del mundo dentro del cerebro. Sólo consiste en tener la consciencia de que el mundo exterior es una memoria externa, accesible de modo inmediato a través de un cambio de atención.

En conclusión, si queremos incorporar en los robots autónomos una cierta capacidad de predecir el resultado de las acciones posibles, necesitamos la mediación de ciertas representaciones abstractas, estructuradas.

1.4. Percepción de alto nivel

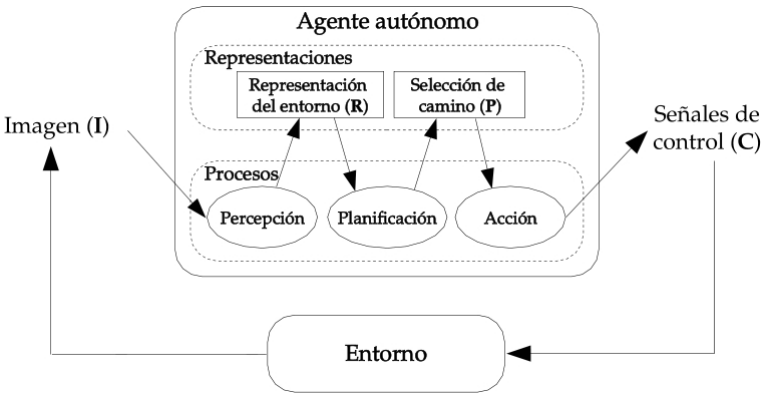

Figura 1.3: Esquema genérico de los sistemas de navegación visual para agentes autónomos.

Una de las características de la inteligencia es la capacidad para establecer analogías entre situaciones aparentemente distintas. Esto no puede resolverse mediante

representaciones estáticas predeterminadas por el diseñador. Hofstadter (1995) sostiene que la clave de la inteligencia está en la percepción de alto nivel, concretamente en la capacidad de construir representaciones estructuradas y reconfigurables dinámicamente. Pero para poder realizar este tipo de percepción el procesamiento ascendente (*bottom-up*), dirigido por los datos, es insuficiente. Es necesario tener en cuenta el contexto, los objetivos, las expectativas, etc. Ese procesamiento debe estar, por tanto, complementado por un flujo descendente (*top-down*) que module el proceso perceptivo, dirigiéndolo a través de la propia representación. *Ver X* es en realidad *ver Y como X*.

Sin embargo, la interpretación constante desde cero de la información sensorial en terminos de representaciones estructuradas, composicionales, es computacionalmente intratable. Se corre el peligro de caer en la explosión combinatoria de las hipótesis candidatas para explicar las observaciones. Las estructuras relacionales conducen frecuentemente a problemas de *graph matching*; un ejemplo típico de *NP-completitud*.

Necesitamos alguna estrategia para luchar contra este tipo de complejidad, especialmente si deseamos construir sistemas capaces de interactuar con el entorno en tiempo real. Hofstadter plantea un modelo computacional basado en la interacción no determinista de numerosos microprocesos elementales, que constantemente construyen, destruyen, reagrupan y reordenan estructuras o hipótesis de interpretacion tentativas. Los conceptos se manejan a un nivel subsimbólico, con numerosas relaciones entre ellos y capacidad de interactuar con conceptos circundantes. La vía de escape ante la explosión combinatoria es la llamada **presión cognitiva**, consistente en considerar simultáneamente diferentes hipótesis con una urgencia que depende de los hallazgos obtenidos en el propio proceso perceptual. Para tratar todas estas alternativas se propone utilizar una ejecución probabilística desviada racionalmente, mejor que una totalmente determinista que podría dejar caminos cerrados desde el principio.

Algunos sistemas inspirados en esta aproximación han tenido éxito en microdominios de cierta complejidad. Casi todos los estudios relacionados, sin embargo, se despreocupan abiertamente de la percepcion a más bajo nivel. La naturaleza inherentemente distribuida de la propuesta, además, no se adapta demasiado bien al paradigma computacional secuencial, por lo que, en principio, no se pueden satisfacer fácilmente las restricciones de operacion en tiempo real. Como veremos a continuación, este proyecto describe un intento de aplicación de las ideas discutidas en este apartado

a un dominio precisamente caracterizado por estos dos problemas.

Capítulo 2
Objetivos

Frasecita.
Autor.

Una vez presentado el contexto bajo el cual se engloba nuestro proyecto, entraremos a detallar los objetivos de éste, así como los requisitos marcados.

2.1. Motivación

Nuestro objetivo es construir un sistema completo de percepción visual de alto nivel, integrado en un robot que se moverá en un entorno real parcialmente estructurado, concretamente el interior de edificios. El comportamiento del sistema deberá obedecer a las propiedades estructurales locales percibidas en el entorno, en contraste con aproximaciones de naturaleza más reactiva frente a los estímulos inmediatos. Para conseguirlo, proponemos una arquitectura de procesamiento inspirada en la interacción de los flujos perceptivos ascendentes y descendentes, al estilo de la propuesta de Hofstadter, pero simplificada y adaptada a las restricciones impuestas por un agente autónomo.

2.1.1. Percepción visual

Nos centraremos en la percepción visual por varios motivos. El primero, y quizás el más importante, es que se trata de la modalidad sensorial más potente en las especies naturales. El escaso esfuerzo consciente que el ser humano emplea en obtener una información detallada de su entorno a partir de las señales luminosas que llegan a su cerebro, puede resultar engañoso a la hora de evaluar la extrema complejidad intrínseca del proceso. Se trata, además, de un campo de investigacion muy abierto y con soluciones satisfactorias sólo en dominios muy restringidos. Incluso en tareas relativamente bien definidas de reconocimiento o clasificacion de objetos en condiciones

más o menos controladas, se presentan multitud de dificultades. La enorme variedad de situaciones que se pueden presentar (entornos cambiantes, condiciones ambientales no controladas, iluminacion variable, reflejos, etc.) impide atacar el problema mediante técnicas de reconocimiento estadístico y requiere la mediación de conceptos de alto nivel.

La visión tiene también la ventaja de que las propiedades del sensor se comprenden bien gracias a la reciente aplicación de la geometría proyectiva a la disciplina. Bajo condiciones muy poco restrictivas, es posible inferir la estructura espacial del entorno a partir de múltiples imágenes de éste. Sin embargo, el extraordinario avance experimentado en los últimos años por las técnicas de reconstrucción tridimensional no ha tenido una respuesta comparable en el desarrollo de métodos adaptados al tiempo real. Muchos de los métodos requieren procesamiento *off-line*, donde primero se adquiere la secuencia completa de imágenes para después tratarlas sin ninguna clase de restricción temporal. Esta metodología resulta inviable en la construcción de sistemas autónomos, donde nos vemos obligados a realizar el procesamiento de la información visual en el mismo momento en que es adquirida.

La estrategia de reconstrucción más utilizada por este formalismo presenta la peculiaridad de que desaprovecha en primera instancia gran parte de la información de coherencia espacial de la imagen. Se parte de características aisladas (normalmente esquinas o segmentos extraídos de la secuencia de imágenes), para rellenar posteriormente la representación obtenida hasta conseguir una reconstrucción densa de la escena. Pero sobre modelos de este estilo no es fácil inferir aspectos cualitativos de su estructura, y estos pueden ser importantes para tomar decisiones no puramente reactivas en aplicaciones de navegación visual como la que nos ocupa.

La alternativa que estudiaremos en este trabajo, por ello, da en cierto modo la vuelta a este paradigma. Mediante procesos de interpretación que tienen en cuenta la consistencia global de la imagen trataremos de inferir previamente las propiedades tridimensionales de la escena, para después corroborarlas o refutarlas con el movimiento a partir de las restricciones multivista. En el ámbito de la robótica móvil, pensamos que es preferible extraer inicialmente la estructura 3D a gran escala para deducir a partir de ahí sus posibles detalles, que fabricar una copia detallada de la realidad para posteriormente interpretarla.

2.1.2. Enfoque ingenieril

A veces resulta enriquecedor para el investigador en percepción artificial realizar una suerte de *ingeniería inversa* sobre los mecanismos desarrollados por la naturaleza. Pero los detalles de implementación no son tan importantes como el logro del objetivo final. Se trata de averiguar las propiedades esenciales de la solución, más que los accidentes de implementación. De hecho, a menudo hay diferencias sustanciales entre los mecanismos perceptivos de las distintas especies, dependiendo de las condiciones evolutivas en cada caso. Por supuesto, es lícito emplear la solución natural como inspiración en los sistemas artificiales, pero tampoco debe suponer un límite si, en determinados aspectos, la implementación concreta adopta un enfoque abiertamente ingenieril.

2.1.3. Estructura del espacio

Como argumentamos anteriormente, las representaciones abstractas constituyen el ingrediente fundamental en la predicción eficiente de los estímulos. Pero los conceptos elementales que forman la base de estas estructuras no se pueden deducir únicamente de la dinámica del espacio sensorial. Se plantea entonces una decisión de diseño de gran importancia. ¿Cuáles son los componentes elementales adecuados, o conceptos primitivos, sobre los que se construyen las representaciones?

En los sistemas biológicos un proceso de prueba y error evolutivo ha contribuido a generar los mecanismos que realizan este trabajo. Las soluciones obtenidas no son necesariamente óptimas en algunos sentidos, pero esencialmente resuelven el problema planteado de modo satisfactorio. En el diseño de sistemas artificiales, no obstante, es evidente de que no estamos todavía en condiciones de automatizar esa etapa. De alguna forma, el agente autónomo debe ser provisto de algún tipo de conocimiento previo, innato, una cierta expectativa del tipo de situaciones que puede encontrar en su mundo.

Afortunadamente, el criterio propuesto en la sección 1.2 nos permite adoptar en realidad cualquier concepto elemental, con la única condición de que mediante su organización estructural se consiga comprender, en el sentido predictivo, el mundo externo. Es evidente que si las regularidades básicas elegidas son frecuentes en el entorno, podrán conseguirse eficientemente representaciones concisas y de gran capacidad predictiva (por ejemplo, a más largo plazo). En caso contrario serán más complejas, costosas de mantener y tendrán menor capacidad de anticipación. En

cualquier caso, para que la elección resulte aceptable, el comportamiento del agente no debería sufrir una degradación catastrófica ante las ausencias, incluso moderadamente frecuentes, de las regularidades esperadas en su dominio de actuación.

2.1.4. Acción y percepción

La percepción está directamente acoplada con las posibilidades de actuación del agente. No en vano, deseamos percibir estructuralmente el entorno para tomar decisiones de movimiento sobre éste. Pero observando el proceso en sentido contrario, la propia acción revierte también de modo significativo en los procesos perceptivos. La interacción dinámica con el medio puede servir para corroborar o refutar las hipótesis de estructuración que sobre él se lanzan inicialmente. Percibir, de hecho, es la mayor parte del tiempo comprobar que todo sigue en su sitio, como debería estar y, sólo esporádicamente, tener capacidad de reacción frente a cambios no esperados o lugares en principio desconocidos.

De alguna manera, esta idea puede ayudar a resolver el problema de la explosión combinatoria que surgiría al tener que evaluar todas las posibilidades de interpretación a cada instante. En lugar de ello, simplemente tendremos que evaluar aquel subconjunto de la masa sensorial que no queda explicado por lo percibido anteriormente. Acción y percepción, pues, se alían en un bucle de sintonía fina que evita la constante construcción de estructuras nuevas, al tiempo que añade la consistencia y continuidad necesarias para un comportamiento eficaz. Los robots dotados de cámaras son, en este sentido, un tipo de sistema artificial muy apropiado para estudiar esta integración.

2.2. Objetivos

El objetivo principal de este proyecto es que un robot pueda deambular en un entorno de oficinas, usando como único sensor una sola cámara situada encima de éste.

Se plantean ciertos problemas a la hora de generar este comportamiento, que hay que superar para conseguir el objetivo marcado. Como objetivo global que se persigue es la navegación del robot Pioneer por un entorno semi-estructurado, como son los pasillos del Departamental de la Universidad Rey Juan Carlos, sin colisionar con ningún obstáculo. Este objetivo global desglosado seria:

- 1. Desarrollo de un algoritmo capaz de obtener los bordes inferiores de una imagen de manera robusta.

- 2. Obtener una estimación de profundidad de los obstáculos que rodean al robot, en una representación local más simple que la captada por la cámara.
- 3. Acumulación de conocimiento a través del movimiento. La interpretación local de cada escena se debe ir acumulando en una representación a largo plazo actualizada constantemente por la odometría. Esta representación podría ser una reconstrucción euclídea completa, con las situaciones tipo reconocidas superpuestas en los lugares correspondientes e interconectadas entre sí (mapa euclídeo-topológico).
- 4. Experimentos. Para estudiar y analizar el comportamiento del robot ante diferentes situaciones reales. Se emplearán diferentes visualizadores gráficos que nos reportarán mucha información del comportamiento de los algoritmos aplicados sobre el robot real.

Todas estas características deben integrarse en un prototipo funcional desarrollado sobre hardware estándar. Los algoritmos y soluciones propuestas para la implementacion de los distintos procesos de percepción visual se adaptarán para ejecutarse eficientemente sobre este hardware, en nuestro caso ordenadores personales basados en microprocesadores convencionales. La plataforma de evaluación de la arquitectura será un robot móvil comercial, de la gama media/baja, dotado de un equipo de computación como el descrito, y conectado con el exterior a través de enlaces inalámbricos.

2.3. Requisitos

El robot operará en entornos estructurados de interiores que son, a priori, desconocidos.

Creemos que el dominio es lo suficientemente variado como para plantear un desafío interesante. Deseamos que el robot interprete el entorno en términos de planos, en el sentido de que los perciba y organice de forma coherente tanto espacial como temporalmente. Deberá ser capaz de extrapolar de esta organizacion conceptos naturales del dominio tales como direcciones preferentes, espacios cerrados o situaciones familiares, por ejemplo. En cualquier caso, la hipótesis de entorno planar tampoco prohibe, como veremos, la existencia de otro tipo de objetos estáticos o dinámicos, ajenos al modelo.

Varios de los requisitos básicos para este proyecto son la utilización del lenguaje C para la programación de los algoritmos bajo la plataforma JDE (que será descrita en el capítulo 3), y bajo el sistema operativo Linux.

2.4. Plan de trabajo

El prototipo desarrollado para lograr estos objetivos se asentará sobre tres pilares básicos:

- **1.** Un mecanismo versátil y eficiente de extracción de información del entorno que asegure la potencia expresiva suficiente para atacar los problemas de los niveles posteriores sin afectar a los requerimientos de tiempo real de la navegación.
- **2.** Un paradigma interpretativo de las escenas de entornos estructurados sobre las que navegará el agente autónomo y que, a través de un esquema de generación, seguimiento y corroboracion o refutación de hipótesis conceptuales sobre la estructura de las mismas, permitan un cierto comportamiento deliberativo en la navegación, más allá de las conductas meramente reactivas.
- **3.** Finalmente, el desarrollo de una arquitectura de acción y percepción que, integrando los componentes hardware y software necesarios en un esquema suficientemente modular y flexible, permita el funcionamiento robusto y eficiente del prototipo sobre entornos estructurados reales, mostrando las deseadas características de autonomía, anticipación y continuidad perceptiva en la navegación.

2.5. Arquitectura global del sistema de percepción

Para cumplir los objetivos anteriores se propone una arquitectura global del sistema de percepción como la mostrada en el diagrama de bloques de la siguiente figura:

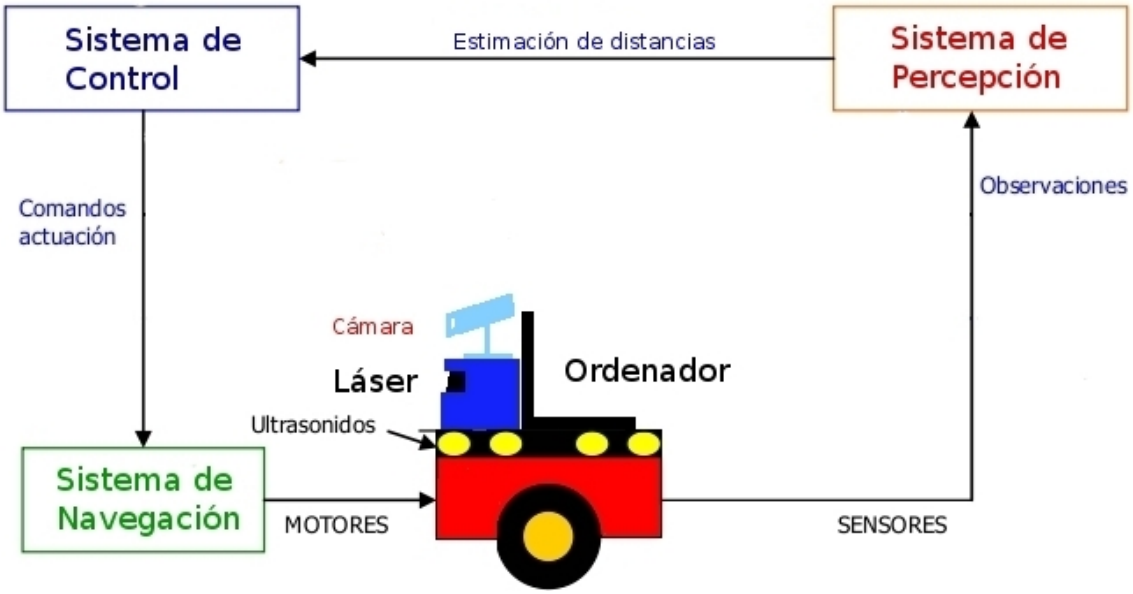

Figura 2.1: Arquitectura global del sistema de percepción.

2.6. Metodología

Para llevar a cabo el proceso software de este proyecto nos hemos basado en un modelo de desarrollo software *incremental*. Dado que se trata de un proyecto que abarca un amplio enfoque estructural, decidimos dividir las distintas partes que conforman el mismo en sucesivos incrementos representativos de cada parte; de este modo, tanto el desarrollo (visto bajo un enfoque global), como las entregas semanales al tutor corresponden a incrementos cuya funcionalidad cumple con los objetivos marcados la semana previa.

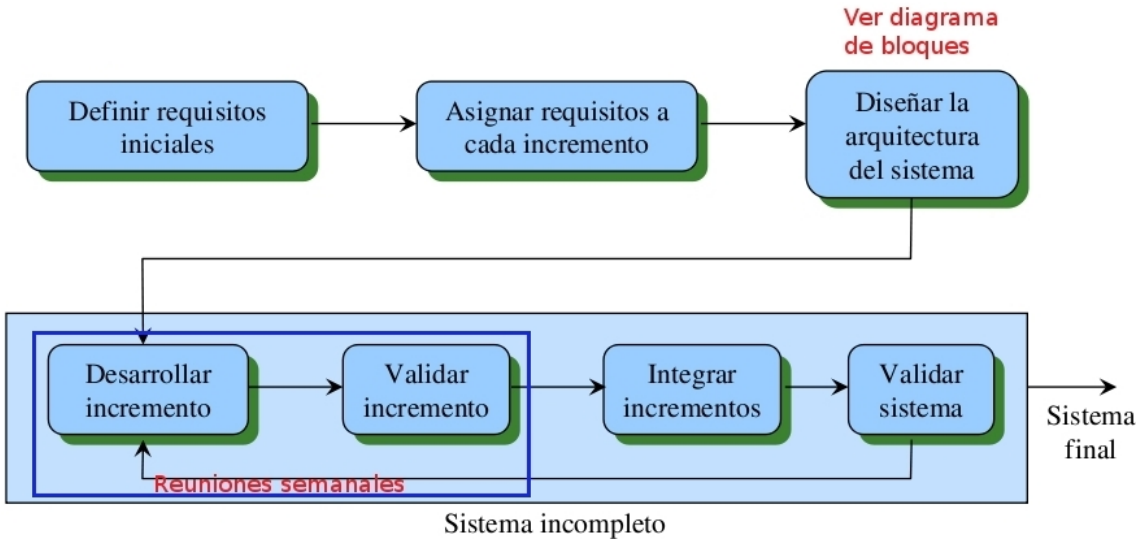

Figura 2.2: Modelo incremental de desarrollo software.

Obviamente, los requisitos de prioridad más alta se incluyen en los incrementos más tempranos. De forma que, cuando el desarrollo de un incremento comienza, sus requisitos son fijos; mientras que los requisitos de incrementos posteriores pueden ir siendo discutidos.

Capítulo 3
Plataforma de desarrollo

Frasecita.
Autor.

En este capítulo vamos a explicar la plataforma software sobre la que ha sido desarrollado este proyecto, cuya implementación comentaremos en los siguiente capítulo. Presentaremos también el robot sobre el que se ejecutará, y comentaremos brevemente algunas bibliotecas de apoyo y herramientas útiles que hemos usado.

3.1. Robot Pioneer

El robot que utlizaremos como referencia en este proyecto es el Pioneer 2-DX, fabricado por ActivMedia Robotics (ver siguiente figura). Se trata de una versátil plataforma móvil autónoma, con dos ruedas motoras laterales y una tercera multidireccional (rueda loca), colocada en la parte trasera para lograr la estabilidad del conjunto. Las ruedas motoras están dotadas de encoders de precisión para la medición de los desplazamientos y giros efectuados por el robot. La alta concentración de estos sensores, con 9850 marcas por revolución en cada rueda (correspondientes a unas 19 marcas por mm, aproximadamente), proporciona un grado de precisión en la odometría que se supondría más que suficiente para las tareas de autocalibración y anticipación en la percepción descritas en el capítulo previo (2). La plataforma dispone también de un conjunto de 8 sensores de ultrasonidos (cubriendo un rango frontal de 180º) para la detección de obstáculos cercanos (a distancias entre 10 y 500 cm), y que funcionan a una frecuencia de unos 25 Hz cada uno. El conjunto está gobernado por un microcontrolador Siemens 88C166 a 20 MHz, con 64 KB de memoria, organizados en 32 KB de flash-ROM para almacenar el microkernel del sistema operativo que controla todos los sensores y actuadores, y otros 32 KB de RAM dinámica para los datos de operación.

Figura 3.1: Robot Pioneer 2-DX sobre el que se centra el proyecto.

La carrocería tiene 33 cm de ancho por 44 de largo y 22 de altura, sin incluir el ordenador portátil que lleva a bordo, así como el sensor láser (que comentaremos en el siguiente apartado). A pesar de lo reducido de las dimensiones comentadas, que dotan al Pioneer de una gran maniobrabilidad incluso en espacios pequeños, la situación ligeramente descentrada de las ruedas permite un diámetro de giro mínimo sobre sí mismo de 52 cm. En cuanto a las características energéticas, estos modelos operan con tres baterías recargables de 12V que le dan una autonomía de funcionamiento de aproximadamente una hora, dependiendo de la cantidad de dispositivos conectados (cámara, PC, sensores láser, equipos de comunicaciones, etc.). El peso total del conjunto es de 9 Kg, aunque puede soportar hasta 20 Kg de carga útil adicional.

La comunicación local entre el PC y el microcontrolador se realiza a través de un puerto serie RS-232. El ordenador portátil está conectado a la red exterior mediante un enlace inalámbrico, con una tarjeta de red 802.11 que le proporciona una velocidad de 11Mbps en sus comunicaciones. De esta forma el programa de control puede correr a bordo del portátil o en cualquier otro ordenador mediante la comunicación wi-fi. Esta última característica es posible gracias a la arquitectura JDE que nos permite una

comunicación mediante el modelo cliente-servidor.

En nuestro proyecto necesitamos, además, información de posición *(x,y,theta)*. Los encoders cuentan las vueltas que dan las ruedas del robot y trabajan con una resolución en mm/sg. Para sacar información de posición a partir de las vueltas contadas es necesario realizar una conversión. Adicionalmente, también le hemos añadido una cámara *i-Sight* (de *Apple*) en color para la captura de imágenes. Está conectada al portátil por el puerto *firewire*. Esta cámara presenta una resolución VGA pudiendo trabajar a una frecuencia de 30Hz (fps).

Una descripcion más detallada de las características del Pioneer 2 y sus distintos componentes puede encontrarse en el manual de descripción del hardware del robot, cuya referencia se proporciona en la bibliografía ([ActivMedia, 2002]).

3.2. Arquitectura. La plataforma JDE

Jdec (Jerarquía Dinámica de Esquemas) es una plataforma desarrollada íntegramente en la URJC ([Cañas Plaza *et al.*, 2007]) para facilitar la programación de robots y de aplicaciones relacionadas con la visión artificial, así como de domótica. Su origen se remonta al año 1997 como fruto de una tesis doctoral ([Cañas Plaza, 2003]). Desde entonces ha ido creciendo año tras año, con nuevas funcionalidades. Este proyecto se ha desarrollado sobre la versión 4.2 de jdec.

Esta plataforma ofrece, en primer lugar, el acceso a los sensores del robot en forma de variables que las aplicaciones leen (*variables perceptivas*), y el acceso a los actuadores como variables que las aplicaciones escriben (*variables de actuación*). En segundo lugar, proporciona un modelo basado en comportamientos concurrentes, llamados *esquemas*, para construir las aplicaciones robóticas.

La ejecución simultánea de varios esquemas dan lugar a un comportamiento. Existen esquemas de distintos tipos: *(i) de servicio*, que se encargan de las comunicaciones recogiendo información de sensores y enviando órdenes a los actuadores; *(ii) perceptivos*, que se encargan de producir y almacenar información sensorial o elaborada por otros esquemas; y *(iii) de actuación*, que son los que toman decisiones sobre motores o la activación de esquemas de niveles inferiores a partir de la información que generan los esquemas perceptivos. Los esquemas pueden organizarse en niveles estableciendo

una jerarquía entre esquemas padre y esquemas hijo siendo el padre el que pueda activar y desactivar a los esquemas hijo. En este proyecto se ha diseñado y programado un esquema de percepción, correspondiente al sistema de percepción sobre el que se fundamenta este trabajo (*frontera*) y uno de servicio (*fronteragui*).

Por último comentar que esta plataforma ofrece, además, soporte para el robot Pioneer con cámaras, para el simulador Stage y la posibilidad de conectarse remotamente a ellos manteniendo el acceso a través de variables. La plataforma también resuelve las necesidades de interfaz gráfica de las aplicaciones, típicamente empleadas para depuración.

Así, nuestro diseño software quedaría establecido del siguiente modo:

- **Robot real.** Las variables representan de forma directa a los sensores y actuadores del robot.
- **Servidores.** Existen dos servidores distintos que proporcionan acceso remoto a los sensores y actuadores, ofreciendo así funcionalidad a los clientes a través de una API de mensajes.

  Player. El servidor de *Player* nos proporciona el acceso a los motores, láser, y sónar.

  Firewire. Se encarga de los sensores de imagen.

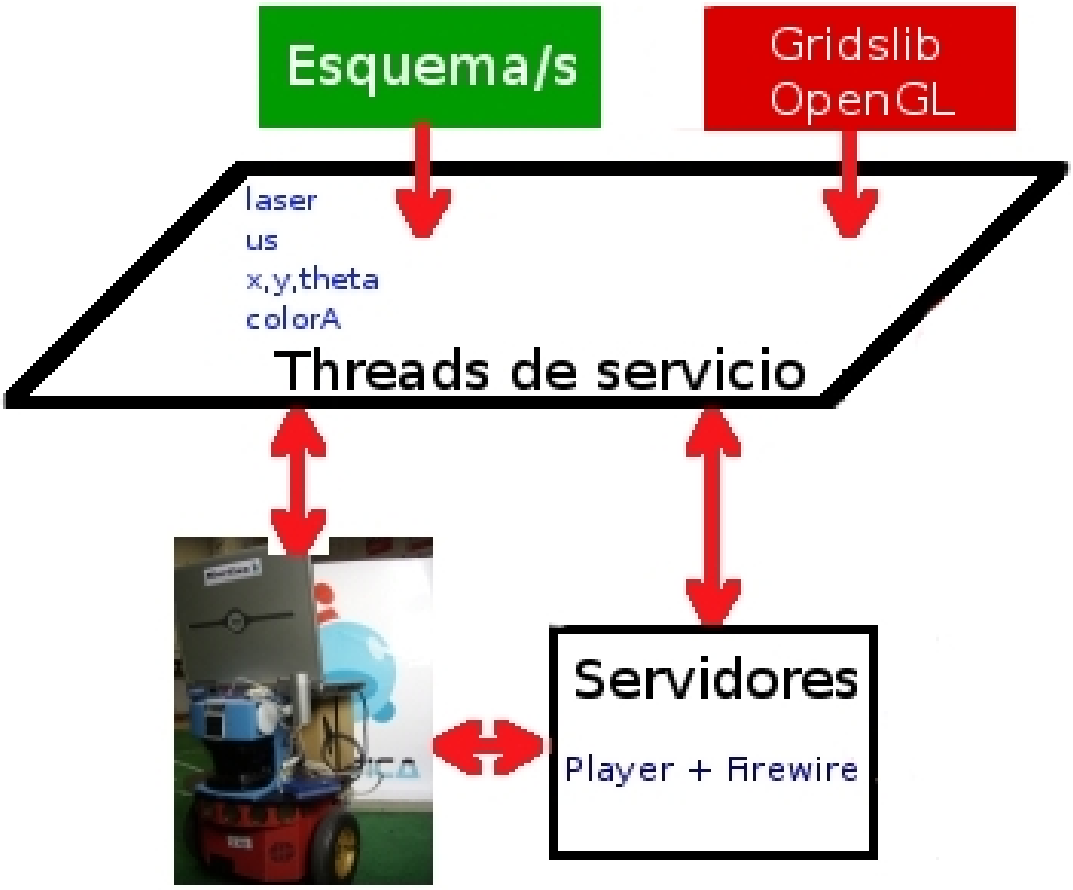

Figura 3.2: Plataforma de desarrollo del proyecto.

3.3. Bibliotecas auxiliares. Gridslib, Progeo, XForms y OpenGL

La librería **Gridslib** nos permite crear y manejar rejillas de ocupación. Esta biblioteca implementa diversas técnicas de construcción de mapas como la regla de Bayes, la regla Dempster-Shafer, rejillas histográmicas o rejillas borrosas. Esta biblioteca genera una rejilla cuadrada con celdas regulares de cierto tamaño. Estos tamaños son especificados en un archivo de configuración que la biblioteca se encarga de leer.

En nuestro proyecto esta biblioteca nos va servir para representar el mapa del escenario. Cada celda de la rejilla almacenará un valor que representará un punto del espacio. Estos valores pueden representar espacios vacíos, paredes, obstáculos. Además podemos asociarle un cierto valor temporal, lo que nos permitirá capacitar al robot de memoria *olvidadiza*, para no acumular ruido excesivo.

La biblioteca **Progeo** ha sido desarrollada en la Universidad Rey Juan Carlos ([Cañas Plaza *et al.*, 2008]), y se basa en las directrices de geometría proyectiva aplicadas por Richard Hartley y Andrew Zisserman ([Hartley y Zisserman, 2000]). Utiliza el *modelo de cámara Pinhole* (Figura 4.2) para las operaciones de proyección de puntos 3D. Aunque no es el único tipo, el modelo Pinhole es el modelo de cámara más común.

Basándose en este modelo de cámara, Progeo, como biblioteca informática de programación, define los tipos de datos y funciones apropiadas para poder realizar los cálculos geométricos que se necesiten. La biblioteca está programada en lenguaje C, permitiendo su integración en proyectos basados en este lenguaje o también en proyectos de lenguaje C++ (independientemente del sistema operativo y entorno de programación del proyecto).

La biblioteca **XForms**, basada en la librería Xlib, es utilizada por el esquema de servicio *fronteragui* utilizado en este proyecto. Nos permite crear una interfaz gráfica con la que visualizar y depurar los resultados que vamos obteniendo a medida que avanzamos con los distintos algoritmos. Esta biblioteca proporciona objetos gráficos como botones, diales, barras de desplazamiento, menús, canvas para visualización de OpenGL (que a continuación detallaremos), etc. para la creación de ventanas en sistemas X Window.

Por último, la librería gráfica **OpenGL** nos va a permitir visualizar de forma gráfica los algoritmos que implementamos. Nos será muy útil como método de depuración. Esta biblioteca trabaja directamente con el procesador gráfico, con lo que la CPU puede ser aprovechada en su totalidad para llevar a cabo los distintos algoritmos.

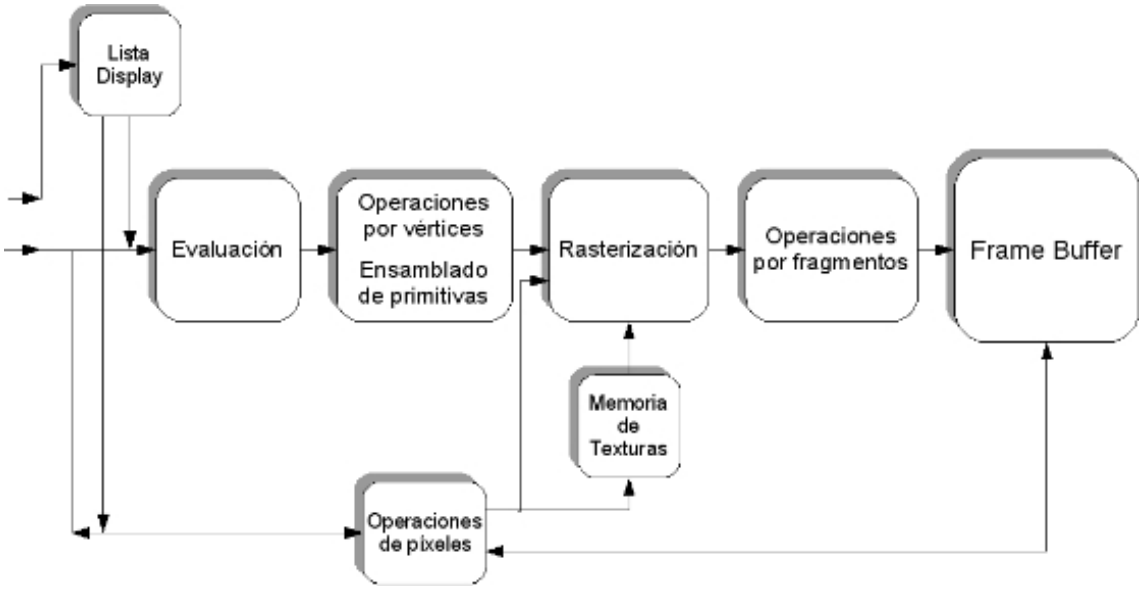

Figura 3.3: Pipeline de OpenGL.

Capítulo 4
Descripción informática

Frasecita.
Autor.

En este capítulo se explica el diseño con esquemas que hemos realizado para solucionar el problema planteado, satisfaciendo los objetivos y requisitos propuestos en el capitulo 2 e implementando éste sobre la plataforma descrita en el capitulo anterior.

Comenzaremos con una aproximación a la solución propuesta, para pasar a describir los esquemas que forman parte de la solución desarrollada.

4.1. Contexto

En los sistemas autónomos que interactúan con el entorno guiados unicamente a partir de la información visual, es fundamental disponer de mecanismos para extraer algun tipo de información estructural de la escena a partir de las imágenes obtenidas. En los ultimos años, la investigación en visión por computador ha experimentado extraordinarios avances en el campo de la reconstrucción tridimensional de escenas a partir de una serie de imágenes tomadas desde distintos puntos de vista. Podemos suponer que las imágenes se toman simultáneamente desde múltiples cámaras, o bien sólo mediante una que se mueve a través de la escena, posiblemente cambiando durante la secuencia su configuración interna (por ejemplo, la distancia focal). Para escenas estáticas, no hay diferencias conceptuales importantes entre ambos casos a la hora de resolver el problema. En general, no obstante, nosotros nos centraremos en el segundo, puesto que la plataforma robótica móvil con la que trabajaremos es monocular, es decir, utiliza una única cámara como sensor óptico.

Las soluciones propuestas para estos problemas están basadas en la aplicación directa de la **geometría proyectiva**, una herramienta matemática que se ajusta muy bien a este campo de la visión artificial ([Hartley y Zisserman, 2000]). Dentro de

las técnicas de reconstrucción basadas en este formalismo pueden alcanzarse distintos niveles en los tipos de reconstrucciones practicadas. Se suele hablar de reconstrucciones proyectivas, afines, similares o euclídeas según el tipo de invariantes que se conservan en la reconstrucción con respecto a la escena original. Así, una reconstrucción proyectiva conserva sólo propiedades tales como la colinealidad, la convexidad, la incidencia o la tangencia, e invariantes como el *cross-ratio* de distancia entre puntos alineados, por ejemplo. Una afín, de nivel inmediatamente superior, comienza a respetar invariantes más fuertes, como el paralelismo entre líneas, ratios de areas, o la situación de la línea en el infinito. Las reconstrucciones similares, adicionalmente, preservan ya distancias relativas y ángulos absolutos. Finalmente, en el nivel superior de la jerarquía, las reconstrucciones euclídeas logran obtener una escena cuya forma y medidas coinciden completamente con las reales, y son capaces de determinar la situación absoluta de las cámaras que tomaron las imágenes sin ningún tipo de ambigüedad.

En general, es poco habitual encontrar métodos que trabajen con este último tipo de reconstrucciones. Ello es debido a que, como es bien conocido, si no se dispone de algún otro dato externo (como el tamaño real de algún objeto, o la magnitud real del movimiento experimentado por la cámara), es imposible determinar la escala global de la imagen reconstruida. Muchas veces, por tanto, el límite máximo que se puede alcanzar en la jerarquía anteriormente comentada es la realización de reconstrucciones similares. Sin embargo, una reconstrucción similar puede no ser suficiente para un agente físico que interactúa con su entorno puesto que, en definitiva, este necesitará antes o después trabajar con medidas reales si pretende moverse en el mismo con seguridad. Aunque, como veremos, es cierto que pueden realizarse ciertas tareas de navegación utilizando sólo propiedades similares o incluso afines (por ejemplo, navegar centrado a lo largo de un pasillo suficientemente ancho), está claro que la obtención de la escala global de la escena ofrece mayores ventajas dado que, en último término, el lazo de control del robot se cierra en términos de órdenes de movimiento absolutas, en coordenadas del mundo reales.

Haciendo uso de un simple par de imágenes, es perfectamente conocido que partiendo sólo de un número de puntos correspondientes entre las mismas puede practicarse una reconstrucción proyectiva de la escena, aun sin tener información previa de ningún tipo sobre su estructura o la calibración de las cámaras con las que fue tomada ([Hartley y Zisserman, 2000], [Zhang y Faugeras, 1992]). Un modo de eliminar la ambigüedad de la reconstrucción proyectiva, y elevarla hasta una

similar es trabajar con cámaras calibradas con anterioridad. En una aproximación más atractiva, sin embargo, puede pensarse en determinar también de modo automatizado las características internas de las cámaras a partir de múltiples vistas, siempre utilizando sólo correspondencias de puntos o líneas entre imágenes. Este enfoque, denominado de **autocalibración**, es una de las líneas de investigación más activas en este campo, puesto que ofrece una flexibilidad y autonomía mucho mayor que el enfoque precalibrado. Tanto es así que se han desarrollado técnicas, incluso, que resuelven el caso en que la cámara también varía algunos de sus parámetros intrínsecos durante la secuencia.

La autocalibración es de gran interés en los sistemas autónomos, puesto que ofrece la posibilidad de determinar las características de los propios sensores a través de la interacción con el medio. En este capítulo nos centraremos en el problema de calibrar una cámara montada a bordo de un robot, tanto en sus parámetros intrínsecos (dependientes de la cámara) como extrínsecos (situación relativa entre la cámara y la plataforma móvil). Quizás, la ventaja más importante la introducirá el hecho de que, en nuestro caso, podamos controlar y conocer los movimientos realizados por el robot. Aunque, como veremos, en general esto no implica conocer el movimiento concreto realizado por la cámara, esta información *extravisual* será suficiente para lograr reconstrucciones euclídeas del entorno, eliminando todo factor de ambigüedad de escala. Esta última propiedad es muy deseable desde el punto de vista de la navegacion, y será una característica central en la arquitectura de percepción propuesta.

4.2. Cámaras en vehículos móviles

En esta sección describiremos el modelo de cámara utilizado, considerando no solo el clásico proceso proyectivo de formación de imagen, sino haciendo también explícita la posición relativa de la cámara en un vehículo móvil sobre el que podemos controlar el movimiento. Presentaremos, pues, en primer lugar una cámara general, que transforma coordenadas de objetos del mundo (en medidas reales) a coordenadas de imagen (en píxeles). Completamos entonces este esquema ampliándolo con la consideración de que la cámara está fijada en el sistema móvil de modo arbitrario, esto es, en posición general respecto a éste. Por último, estudiaremos también una serie de simplificaciones sucesivas que, en algunos casos, facilitarán las tareas de calibración y reconstrucción, discutiendo su aplicabilidad y viabilidad en distintas situaciones.

4.2.1. Modelo de cámara general

El modelo de cámara que tradicionalmente se utiliza para pasar de coordenadas reales *3D* a coordenadas *2D* pertenecientes a la imagen captada, suele ser el modelo de proyección perspectiva denominado modelo *pin-hole*. En dicho modelo todos los rayos provenientes de un cierto objeto atraviesan un fino agujero para impactar en el sensor imagen. Dado que las lentes no tienen este comportamiento lineal, el modelo *pin-hole* debe ser corregido con un valor de distorsión, es decir, debe ser complementado con parámetros que corrigen su comportamiento ideal y lo acercan, lo más posible, al comportamiento real del objetivo. En la siguiente figura se muestra un ejemplo de proyección perspectiva utilizando el modelo *pin-hole* de un objeto *3D* sobre un plano imagen *2D* mediante funciones *OpenGL*.

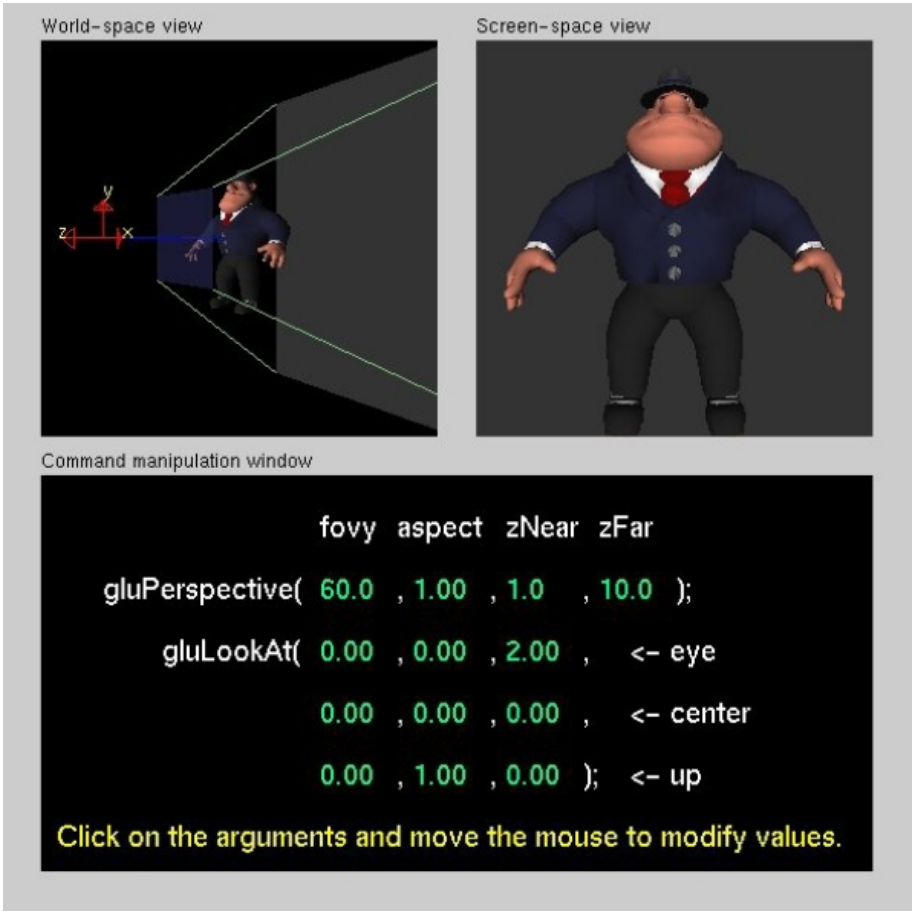

Figura 4.1: Ejemplo de proyección usando el modelo pin-hole en OpenGL (Nate Robins).

El sistema de referencia de la cámara se sitúa en el centro de la proyección, coincidiendo el eje z de dicho sistema con el eje óptico, también denominado eje axial. En esta disposición de ejes, el plano imagen, de coordenadas u,v, se encuentra situado

a una distancia igual a la longitud focal del objetivo de forma perpendicular al eje óptico. La intersección del eje óptico con el plano imagen se denomina punto principal.

El centro de proyección C de la cámara se supone constante pero es a priori desconocido. El plano imagen normalmente se sitúa por delante del centro de proyección C para tener una imagen sin inversión. A continuación se muestra un esquema explicativo del modelo *pin-hole*.

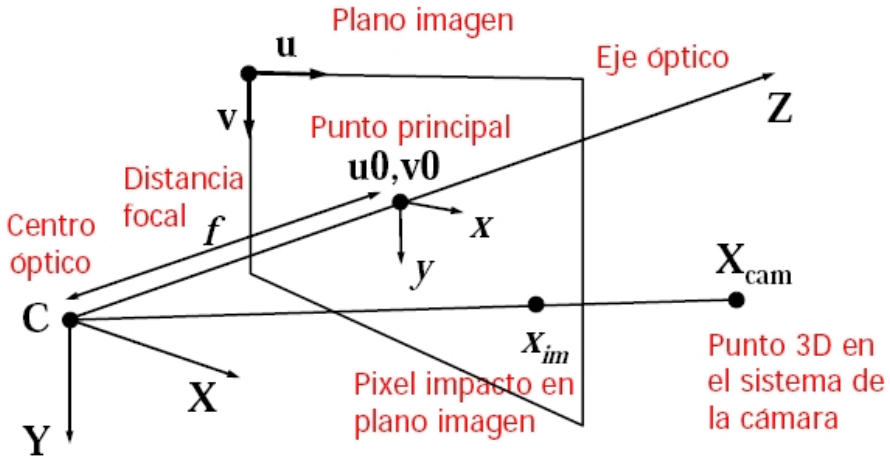

Figura 4.2: Diagrama explicativo del modelo *pin-hole*.

4.2.2. Modelo de cámara rígidamente acoplada a un móvil

Para el problema que nos ocupa resulta interesante hacer explícita la posición de la cámara respecto al sistema móvil, dado que es en este último en el que podemos dar las órdenes de movimiento y tomar los valores de odometría. Supondremos un vehículo holonómico, esto es, que puede girar sobre sí mismo sin necesidad de trasladarse (como es el caso de la plataforma robótica utilizada como base de la arquitectura de percepción propuesta, explicado en el capítulo 3). Esto significa que las órdenes de traslación sobre el plano del suelo y giro (*theta*) pueden proporcionarse por separado. En la figura siguiente se muestra el esquema de posición general de cámara en un robot de estas características.

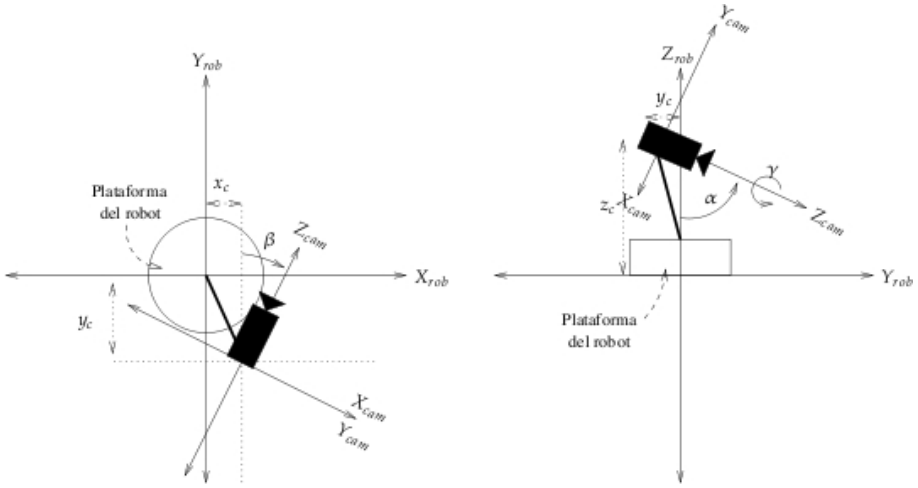

Figura 4.3: Vistas superior y lateral de la situación relativa de la cámara respecto al robot.

4.2.3. Modelos simplificados

En ocasiones, puede ser aceptable realizar algunas restricciones sobre el modelo descrito, con el fin de facilitar los métodos de calibración o reconstrucción que sobre él quieran aplicarse posteriormente. En este apartado describimos algunas de estas simplificaciones, clasificándolas según trabajen sobre los parámetros intrínsecos o extrínsecos de la cámara, y discutiendo sobre las distintas condiciones de aplicabilidad de cada una.

El problema de realizar reconstrucciones a partir de imágenes en movimiento es esencialmente la sensibilidad al ruido. Algunos autores han enfatizado el problema de la baja resolución de las cámaras de computador, frente a las empleadas en fotogrametría, de mayor precisión. En estos últimos dispositivos puede tener más sentido realizar calibraciones exhaustivas. Sin embargo, en muchos ámbitos de aplicación de la visión artificial donde cierto tipo de errores pueden considerarse admisibles, es viable realizar ciertas suposiciones sobre las características de la cámara utilizada que se traducen en condiciones sobre la matriz de ***parámetros intrínsecos***. La razón es que la deformación afín introducida por dicha matriz es quizás demasiado general, puesto que somete a consideración incluso la perpendicularidad de los ejes de coordenadas de la cámara...

Veamos ahora las posibles simplificaciones sobre los ***parámetros extrínsecos*** que pueden resultar de interés. La primera es la que llamaremos hipótesis de *brazo nulo*.

Esta hipótesis supone que $Xc = Yc = 0$, es decir, que la cámara no solo está rígidamente unida al vehículo, sino que además experimenta una traslación de módulo igual a la que mide el robot.

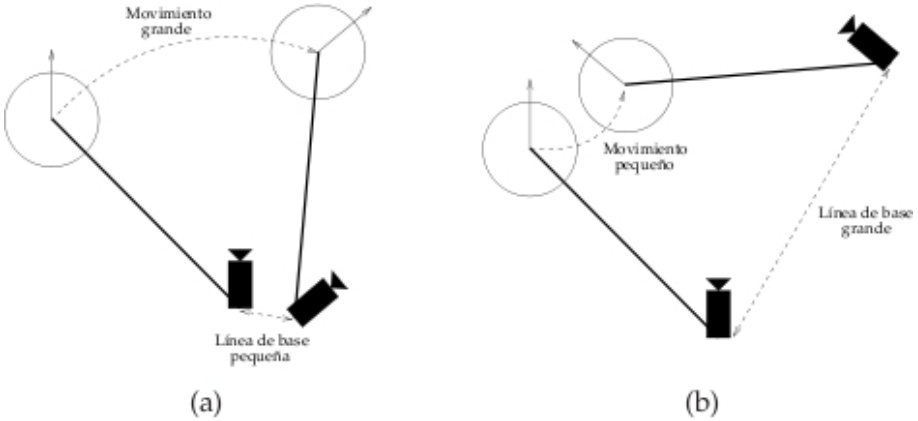

Figura 4.4: Traslación parásita asociada a una cámara montada sobre un robot con brazo no nulo. (a) Un movimiento largo correspondiente a una línea de base corta. (b) Caso contrario.

En segundo lugar, podría también considerarse que la cámara se monta sin ángulo de roll. En estas condiciones, la línea del infinito del plano del suelo aparecería siempre en la imagen como una línea horizontal. Pero en nuestro caso, no haremos esta consideración, puesto que la cámara deberá estar inclinada *ligeramente* hacia el suelo, ya que deberemos detectar las fronteras entre éste y el objeto-obstáculo. La siguiente figura representa la situación de desfase entre el plano de imagen de la cámara y el suelo.

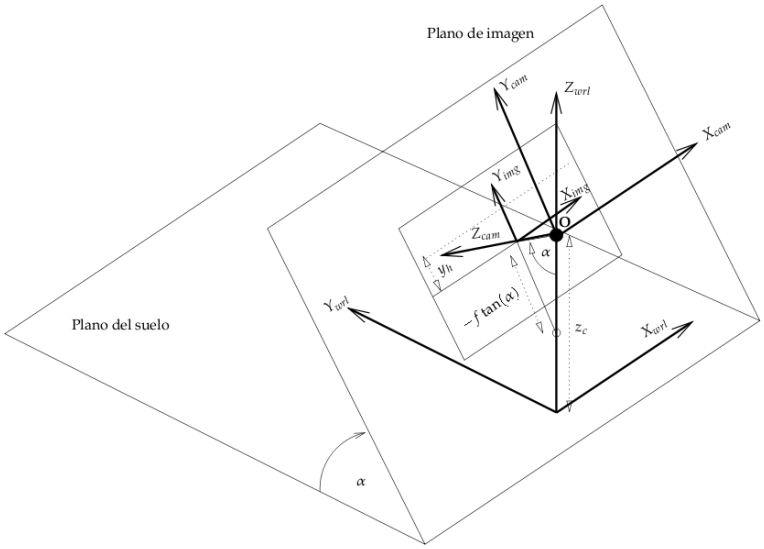

Figura 4.5: Definición del sistema de coordenadas del mundo respecto a la posición de la cámara. El punto sólido indica la posición del centro óptico de la cámara.

Finalmente, un tercer caso que merece la pena estudiar es el de ángulo de pan también nulo. Ello equivale a colocar la cámara mirando en la dirección de avance del robot, sin desfase apreciable entre la dirección de avance de éste y el eje óptico de aquélla.

4.3. Calibración empleada

Una vez visto en la sección anterior el contexto bajo el cual partimos a la hora de calibrar la cámara acoplada al robot, contemplando los problemas con los que nos encontramos y considerando algunas simplificaciones ya citadas, pasaremos a comentar en este apartado la calibración que hemos empleado en el proyecto.

Dado que los parámetros intrínsecos son propios de la cámara y no del movimiento del robot a lo largo de su navegación, éstos los hemos extraído *a priori* empleando para ello el proyecto [Kachach, 2008]. Basándose en la técnica de DLT (*Direct Linear Transformation*), a partir de un patrón de calibración 3D del cual se conoce perfectamente su geometría y la posición de ciertos puntos significativos en él, el algoritmo funciona del siguiente modo:

- Capturar la imagen del patrón 3D.
- Detectar los píxeles donde caen los puntos 3D característicos.

- Construir un sistema de ecuaciones con la información de correspondencia entre ambos.
- Resolver el sistema optimizando la matriz solución y extraer los parámetros de la cámara descomponiendo esta última.

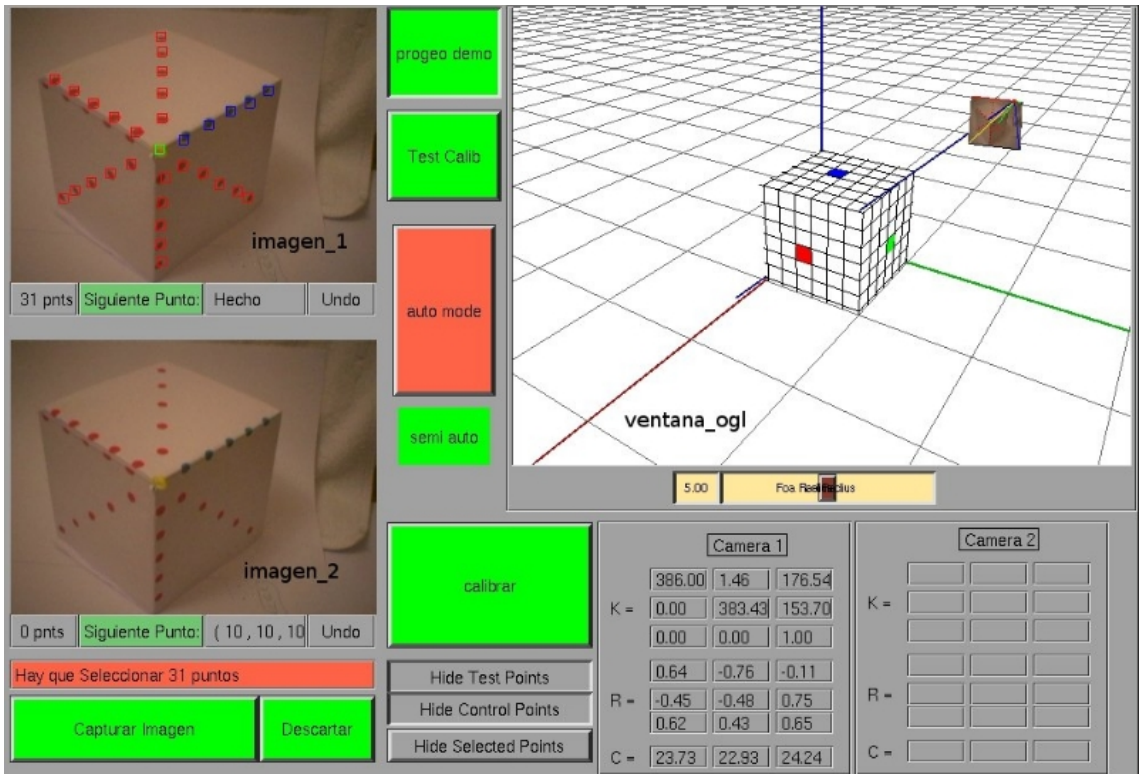

Figura 4.6: Interfaz gráfica del calibrador empleado.

Una vez obtenidos los parámetros intrínsecos, sólo nos quedarían establecer los paráameros extrínsecos (situación de la cámara en el mundo). Se plantea la utilización de la odometría como apoyo natural a la calibración, con el fin de obtener medidas reales de la escena sin ningún tipo de ambigüedad proyectiva, afín o similar en las reconstrucciones practicadas. La información euclídea conseguida resultará de gran utilidad en la interpretación estructural posterior y, en consecuencia, también en la navegación.

En nuestro Pioneer 2 DX, a pesar de su sencillez, el sistema de medición del giro de las ruedas a través de la lectura de encoders puede llegar a ser bastante preciso si se calibran correctamente ciertas características mecánicas de la plataforma, como el radio y posición relativa entre las ruedas, el número de encoders por rueda, etc. Esto es sobre todo cierto para desplazamientos relativamente pequeños, puesto que dado el

carácter inherentemente incremental de este tipo de medición, los errores son también acumulativos. Afortunadamente, las técnicas que propondremos en las secciones posteriores no necesitan desplazamientos largos, por lo que la precisión lograda es más que suficiente si se tiene cuidado en evitar movimientos especialmente mal condicionados para la calibración. En la última sección de este capítulo mostraremos algunos experimentos que corroboran esta última afirmación, al menos para el ámbito de aplicación que nos ocupa.

Los experimentos realizados demuestran la viabilidad del uso de la odometría en la autocalibración, dada la aceptable precisión lograda por las técnicas descritas en tareas relacionadas con la navegación visual. La escasa carga computacional de todas ellas, junto con su adecuación a los entornos estructurados, hacen a la aproximación idónea para ser utilizada *on-line* en la arquitectura de percepción propuesta.

4.4. Diseño general

El objetivo de este proyecto, como ya se ha comentado, es que un robot perciba su entorno de forma totalmente autónoma, utilizando exclusivamente visión monocular, mediante técnicas de visión computacional (mencionadas en las secciones anteriores) y pueda actuar en consecuencia con algún sistema de navegación (con el cual no entraremos en detalle).

La implementación de este proyecto se ha hecho sobre la plataforma *jde.c* y el lenguaje de programación C. Así, el diseño general para el comportamiento buscado se podrá resumir en una hebra o esquema *JDE* que iterativamente procesa las imágenes de la cámara.

Este único esquema, llamado *frontera* se encarga de calcular las estimaciones de distancia del robot a los objetos u obstáculos. Por lo tanto es un esquema perceptivo, que utiliza como entrada las imágenes y puede ofrecer a otro esquema (sistema de navegación) los datos de profundidad obtenidos.

También existe un esquema de servicio *fronteragui* que interactúa con el usuario y desde el cual se activan todas las funionalidades, además de visualizarse en todo momento la situación del robot y las estructuras de datos de los esquemas anteriores.

El trabajo del esquema *frontera* consiste básicamente en un procesamiento monocular; se haya la distancia a los objetos en el campo visual de la cámara, apoyándose en la **hipótesis de suelo** en la que se asume que la parte baja de la imagen es suelo. Es un esquema en el que se realizan múltiples cálculos geométricos necesarios para la traducción de puntos 2D de la imagen a puntos 3D del entorno del robot. Realiza 10 iteraciones por segundo en las que se actualizan las mediciones. Dentro de los ficheros frontera.c y frontera.h se encuentra el código cuya funcionalidad está representada en la siguiente figura.

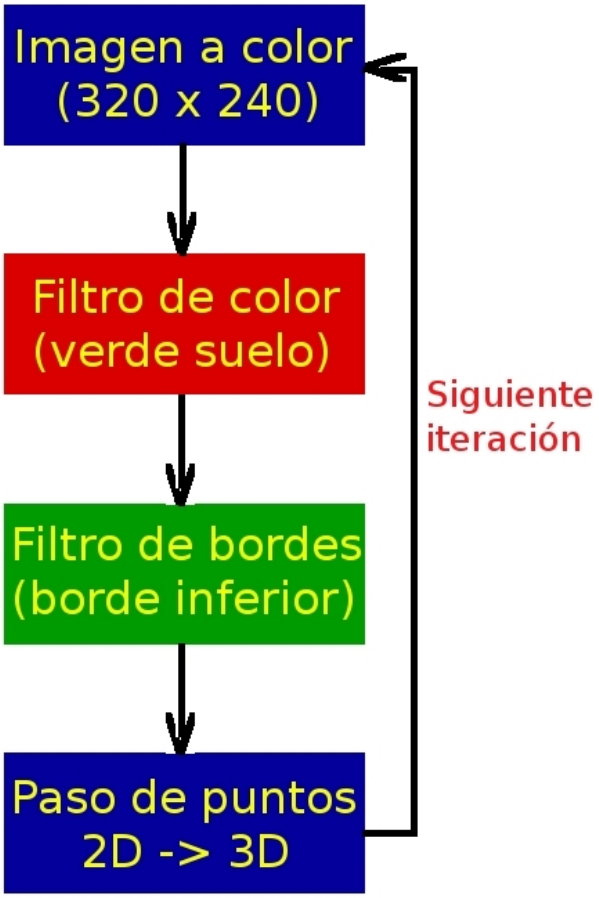

Figura 4.7: Pseudocódigo del esquema frontera.

4.4.1. Procesamiento monocular

Inicialmente se filtra la imagen RGB *colorA*, captada por la cámara, por el color del suelo; en nuestro caso, dado que tenemos la intención de su óptimo funcionamiento sobre el campo de fútbol en el que compiten los robots *Sony Aibo* en la *Robocup* (torneo de fútbol robótico a nivel mundial), será éste el color de filtrado. En la siguiente imagen vemos un ejemplo de este filtrado, aplicado sobre la imagen colorA, dentro del campo

de fútbol robótico.

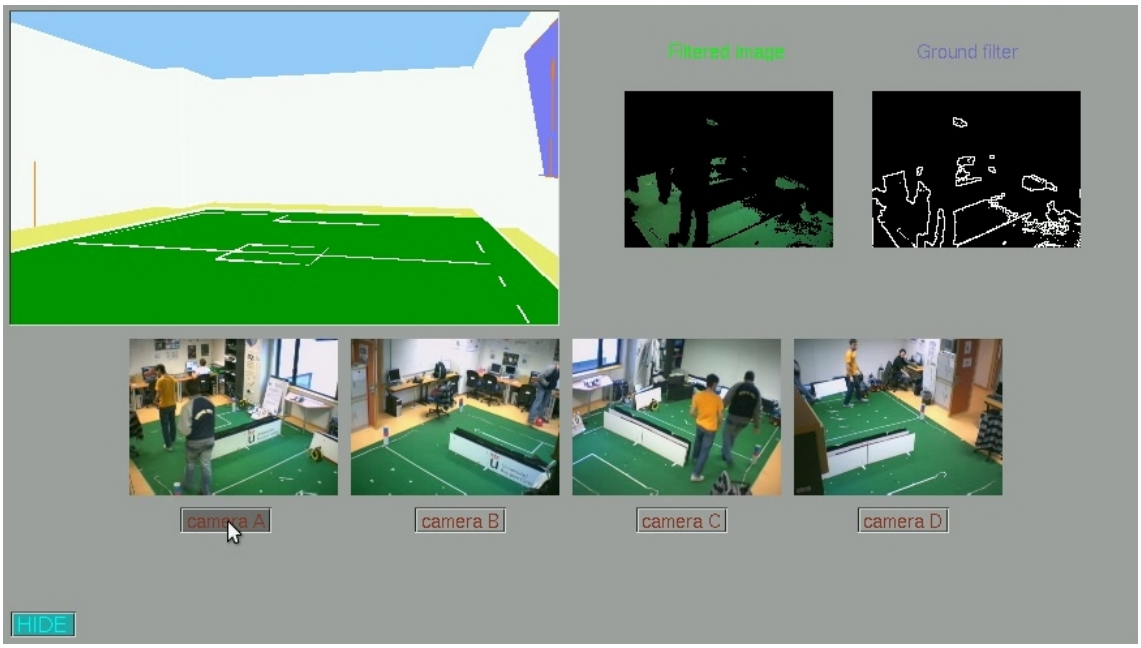

Figura 4.8: Ejemplo de filtrado de color sobre la imagen RGB.

Se ha desarrollado un laboratorio virtual, empleando la biblioteca gráfica *OpenGL*, para realizar depuraciones de forma más intuitiva. Está basado en el Laboratorio de Robótica de la URJC, donde se realizan todos los experimentos. Lo mostramos a continuación:

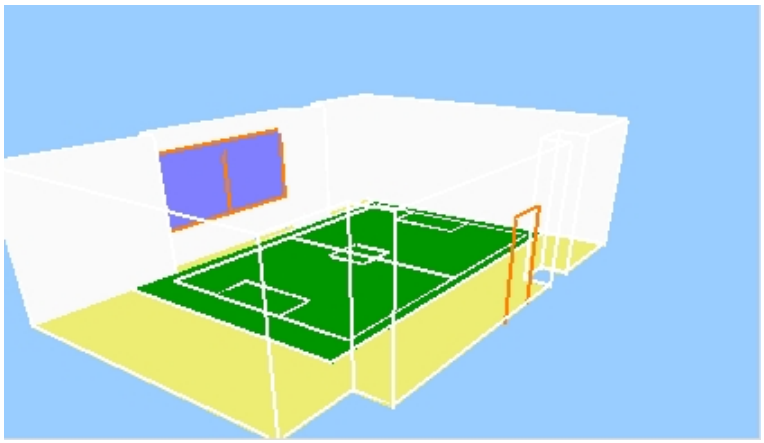

Figura 4.9: Laboratorio virtual.

El hecho de tener las paredes *transparentes* nos facilita la visualización del interior del habitáculo virtual. Basándonos en las propiedades de *Cull-Face*, empleadas hoy día en multitud de videojuegos, podemos definir los planos de forma que el vector normal definido por los vértices que componen a éstos nos de información de si estamos *mirando*

tal plano por delante o por detrás, optimizando así la visualización y el renderizado gráfico.

Pues bien, el proceso de filtrado es necesario para operaciones posteriores sobre la imagen. Se realiza de la forma siguiente:

- Se recorre la imagen píxel a píxel.
- Obtenemos las componentes RGB del píxel actual.
- Pasamos de coordenadas RGB a HSV.
- Filtro de color HSV con las tres componentes antes obtenidas (H, S y V).

El fragmento de código que realiza tal función se muestra a continuación:

```
void drawFilteredImage () {
  int i,c,row,j,k;
  double H, S, V;
  float myR, myG, myB;

  for (k=0; k<SIFNTSC_ROWS*SIFNTSC_COLUMNS; k++) {
    c=k%(SIFNTSC_COLUMNS);
    row=k/(SIFNTSC_COLUMNS);
    j=row*SIFNTSC_COLUMNS+c;

    if (((display_state&DISPLAY_COLORIMAGEA)!=0) && (mycolorA!=NULL)) {
      imagenFiltrada_buf[k*4] = (*mycolorA)[j*3];
      imagenFiltrada_buf[k*4+1] = (*mycolorA)[j*3+1];
      imagenFiltrada_buf[k*4+2] = (*mycolorA)[j*3+2];
    } else { /* no recibimos señal de la cámara */
      imagenFiltrada_buf[k*4]=0;
      imagenFiltrada_buf[k*4+1]=0;
      imagenFiltrada_buf[k*4+2]=0;
    }
    imagenFiltrada_buf[k*4+3]=0; /* bit dummy */

    myR = (float)(unsigned int)(unsigned char)imagenFiltrada_buf[k*4+2];
    myG = (float)(unsigned int)(unsigned char)imagenFiltrada_buf[k*4+1];
    myB = (float)(unsigned int)(unsigned char)imagenFiltrada_buf[k*4+0];

    myHSV = (struct HSV*) RGB2HSV_getHSV ((int)myR,(int)myG,(int)myB); /* pasamos de RGB -> HSV */

    if (!((myHSV->H*DEGTORAD > 1.7017) && (myHSV->H*DEGTORAD < 2.5394) &&
      (myHSV->S > 0.31) && (myHSV->S < 0.7) &&
      (myHSV->V > 77.6) && (myHSV->V < 207.2))) { /* si no pasamos el filtro de campo fútbol */
      imagenFiltrada_buf[k*4]=0;
      imagenFiltrada_buf[k*4+1]=0;
      imagenFiltrada_buf[k*4+2]=0;
    }
  }
}
```

Figura 4.10: Implementación del filtrado de color.

Una vez está la imagen filtrada según el color deseado, se pasa ésta por un filtro de bordes, cuyo funcionamiento es el siguiente:

- Se recorre la imagen filtrada píxel a píxel.
- Si este píxel NO es del color de filtrado, compruebo sus vecinos.
- Si alguno de los vecinos el color del campo, podemos asegurar que el punto actual es un ***punto-frontera***.

Podemos ver este comportamiento en la figura 4.8 donde, junto a la imagen filtrada (parte superior derecha), está la imagen en blanco y negro que representa los puntos frontera encontrados (puntos en blanco).

Pues bien, una vez detectada la discontinuidad del suelo con los objetos, podemos calcular en dimensiones de la imagen estos puntos discontinuos. Pero estos datos en sí no nos sirven para estimar la distancia del robot a los obstáculos. De ahí que nos apoyemos en un modelo de cámara (modelo pin-hole comentado en el apartado 4.2.1), y en una librería que hace uso de este tipo de cámara, como es *Progeo* (ver 3.3).

Utilizando el esquema *calibrador* (4.6), obtendremos los parámetros necesarios para una cámara de este tipo: una matriz K (definida mediante los elementos *k11..k34*) y una matriz de rotación y translación (definida mediante los parámetros *rt11..rt44*), creadas a partir de sus parámetros intrínsecos y extrínsecos.

Con estos datos ya tenemos calibrada nuestra cámara y podremos entonces aplicar la funcionalidad de *Progeo*; en nuestro caso, sólo nos será necesario la función *backproject*. Mediante esta función se pueden obtener las coordenadas 3D de un punto proyectado en varios planos imagen de diferentes cámaras. Para ello, dado un punto 2D contenido en el plano imagen de una cámara concreta, permite obtener la representación 3D de ese punto. Esa representación, junto con el foco de la cámara desde la que se observe, permite trazar un segmento en el espacio que contiene al punto 3D buscado. De esta forma, si se realiza la misma operación para el mismo punto pero desde otra cámara (con otra proyección 2D distinta), es posible hayar un nuevo vector director, que o bien corta o se cruza con el anterior. El punto de cruce en el espacio es precisamente el punto 3D buscado.

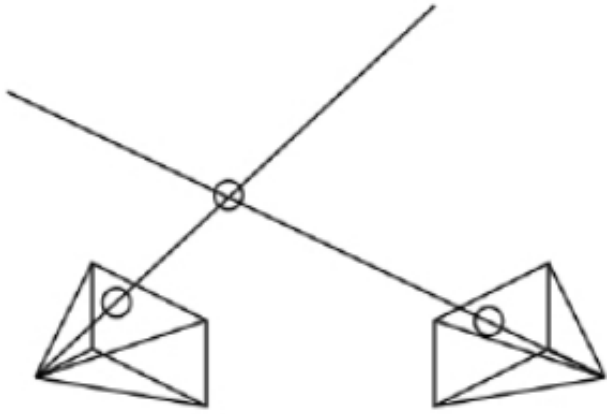

Figura 4.11: Proceso de triangulación.

La cabecera de la función, por tanto, recibe un punto 2D y la cámara concreta del plano imagen como entradas, y devuelve un punto 3D (la representación 3D del punto 2D de entrada):

```
int backproject(HPoint3D *out, HPoint2D in, TPinHoleCamera camera);
```

Pero nosotros, como ya hemos dicho, no tenemos dos cámaras, sino sólo una. Es ahora el momento de utilizar la que hemos llamado ***hipótesis de suelo***. Ésta se basa en que la cámara, situada encima del robot, está inclinada enfocando al suelo. Esto hace que la imagen, observándola de abajo a arriba, se componga de suelo y demás objetos. La discontinuidad entre éstos es la calculada por el filtro de bordes y la que vamos a traducir a coordenadas 3D referentes al robot.

Definimos el plano que contiene el suelo sobre el que está situado el robot. Los píxeles frontera situados en 3D se unen con el centro óptico de la cámara. De este modo, los rayos formados se intersectan con el plano del suelo. El código que realiza tal sutileza es el siguiente:

```
int linePlaneIntersection (HPoint3D A, HPoint3D B, HPoint3D *intersectionPoint) {
  HPoint3D v; /* vector director de la línea A-B */

  v.X = (A.X - B.X);
  v.Y = (A.Y - B.Y);
  v.Z = (A.Z - B.Z);

  /* Intersección con el plano suelo (Z = 0). Ecuaciones paramétricas */
  intersectionPoint->Z = 0; // intersectionPoint->Z = A.Z + t*v.Z => t = (-A.Z / v.Z)
  t = (-A.Z) / (v.Z);

  intersectionPoint->X = A.X + (t*v.X);
  intersectionPoint->Y = A.Y + (t*v.Y);
}
```

Figura 4.12: Intersección del rayo proyectado con el plano suelo.

En la siguiente figura podemos observar cómo a partir de la imagen de entrada (colorA) situada en la parte superior derecha de la imagen realizamos el pertinente filtrado de color, obteniendo los *puntos-frontera* (parte inferior izquierda). Obtenemos la representación en 3D de todos ellos (parte superior izquierda), y unimos con el centro óptico de la cámara (también situado en 3D), obteniendo así los rayos de proyección de los *puntos-frontera* (líneas grisáceas de la imagen). Calculando la intersección de estos rayos con el plano suelo, podemos obtener los puntos en 3D del cubo que se pretende representar (parte inferior derecha).

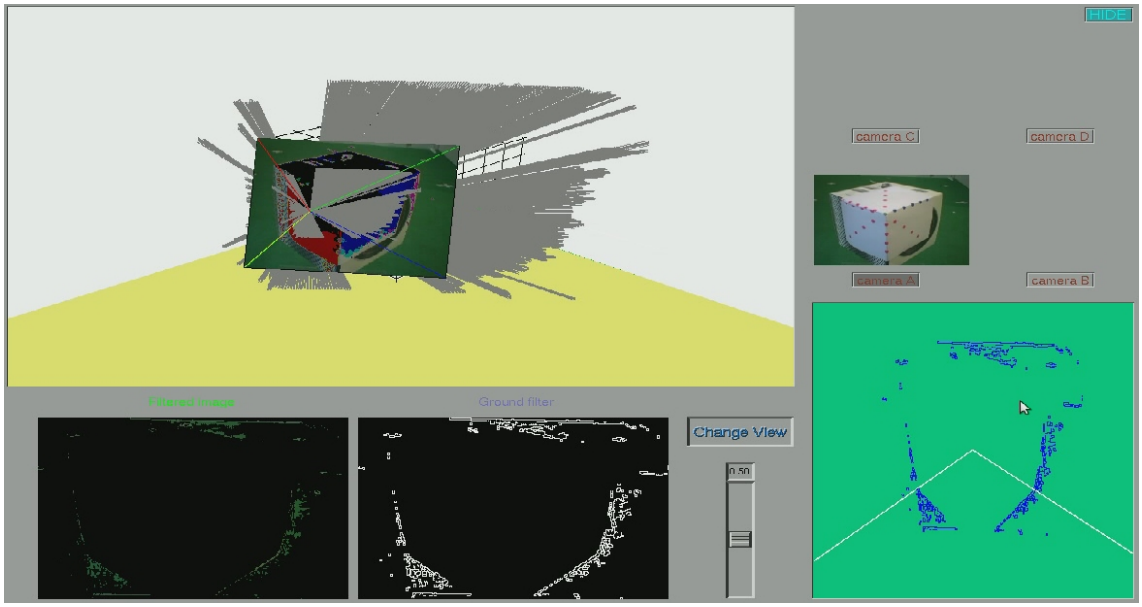

Figura 4.13: Ejemplo de la técnica de proyección e intersección con el plano suelo.

Obviamente, para estimar las distancias hacia los obstáculos (en este caso el cubo), tan sólo nos será necesario los puntos que separan el suelo de los obstáculos. Por tanto, el siguiente paso será extraer sólo estos puntos. Para ello:

- Recorremos la imagen de *puntos-frontera* por columnas.
- Por cada columna, recorro la imagen de abajo a arriba.
- El primer *punto-frontera* obtenido será el punto que separa el suelo del obstáculo.
- Pasamos a la siguiente columna y repetimos el proceso.

En resumen, el proceso total seguido se muestra en la siguiente figura, con un ejemplo ilustrativo.

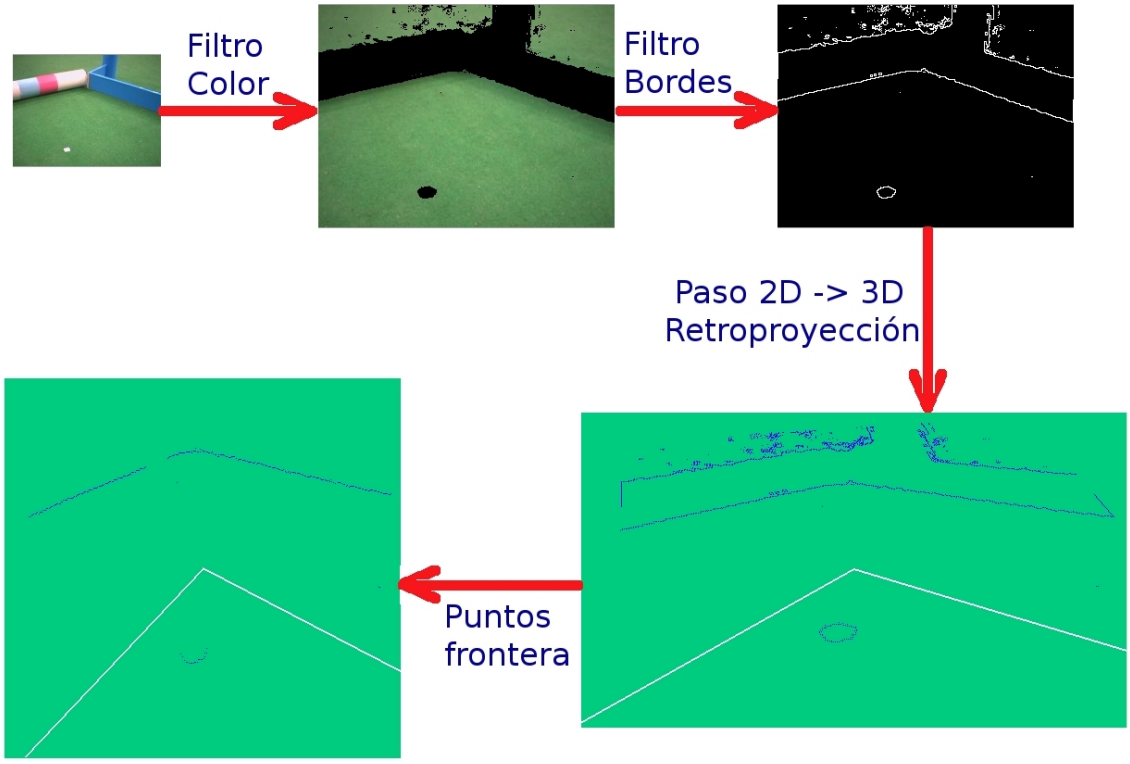

Figura 4.14: Ejemplo del proceso total seguido.

4.5. Experimentos

Capítulo 5
Conclusiones

Bibliografía

[ActivMedia, 2002] Robotics ActivMedia. *Pioneer 2. Operations Manual.* ActivMedia Robotics, 2002.

[Bustos, 1998] Pablo Bustos. *Generación de comportamiento complejo en robots autónomos.* PhD thesis, Universidad Politécnica de Madrid, 1998.

[Camus *et al.*, 1999] T. Camus, D. Coombs, M. Herman, y T. Hong. Real-time single-workstation obstacle avoidance using only wide-field of view divergence. *Journal of Computer Vision Research*, 1999.

[Cañas Plaza *et al.*, 2007] José M. Cañas Plaza, Antonio Pineda, Jesús Ruíz-Ayúcar, José A. Santos, y Javier Martín. *Programación de robots con la plataforma jdec.* Universidad Rey Juan Carlos, 2007.

[Cañas Plaza *et al.*, 2008] José M. Cañas Plaza, Antonio Pineda, Manuel Mendoza, y Víctor Hidalgo. *Biblioteca de geometría proyectiva Progeo.* Universidad Rey Juan Carlos, 2008.

[Cañas Plaza, 2003] José María Cañas Plaza. *Jerarquía Dinámica de Esquemas para la generación de comportamiento autónomo.* PhD thesis, Universidad Politécnica de Madrid, 2003.

[Coianiz y Aste, 1993] T. Coianiz y M. Aste. Improving robot's indoor navigation capabilities by integrating visual, sonar and odometric measurements. *Istituto per la Ricerca Scientifica e Tecnologica, Trento (Italy)*, 1993.

[de Miguel, 2005] Darío de Miguel. Detección automática del color de la piel en imágenes bidimensionales basado en el análisis de regiones. *Proyecto fin de carrera Ing. Tec. Informática de sistemas, Universidad Rey Juan Carlos*, 2005.

[Díaz, 2005] Pedro M. Díaz. *Navegación visual del robot Pioneer.* Proyecto fin de carrera Ing. Téc. Informática Sistemas, Universidad Rey Juan Carlos, 2005.

[García, 2007] Iván García. *Reconstrucción 3D visual con algoritmos evolutivos.* Proyecto fin de carrera Ing. Téc. Informática Sistemas, Universidad Rey Juan Carlos, 2007.

[Gómez, 2002] Víctor M. Gómez. *Comportamiento sigue pared en un robot con visión local.* Proyecto fin de carrera Ing. Téc. Informática Sistemas, Universidad Rey Juan Carlos, 2002.

[Hartley y Zisserman, 2000] R. Hartley y A. Zisserman. Multiple view geometry in computer vision. *Cambridge University Press, Cambridge (UK)*, 2000.

[Kachach, 2008] Redouane Kachach. *Calibración automática de cámaras en la plataforma jdec.* Proyecto fin de carrera Ing. Informática, Universidad Rey Juan Carlos, 2008.

[López, 2003] Pedro E. López. *Una Arquitectura Eficiente de Percepción de Alto Nivel: Navegación Visual para Robots Autónomos en Entornos Estructurados.* PhD thesis, Universidad de Murcia, 2003.

[Mallot *et al.*, 1997] H. Mallot, M. Franz, B. Schölkopf, y H. Bülthoff. The viewgraph approach to visual navigation and spatial memory. *Proceedings of the 7th International Conference on Artificial Neural Networks, Lausanne (Switzerland)*, 1997.

[Montiel y Zisserman, 2001] J. Montiel y A. Zisserman. Automated architectural acquisition from a camera undergoing planar motion. *Proceedings of the International Symposium on Virtual and Augmented Architecture, Dublin (Ireland)*, 2001.

[Moravec, 1996] H. Moravec. Robot spatial perception by stereoscopic vision and 3d evidence grids. *Informe Técnico No. CMU-RI-TR-96-34, Robotics Institute, Carnegie Mellon University, Pittsburgh (USA)*, 1996.

[Riseman *et al.*, 1997] E. Riseman, A. Hanson, J. Beveridge, R. Kumar, y H. Sawhney. Landmark-based navigation and the acquisition of environmental models. *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles, Aloimonos*, 1997.

[Ruf y Horaud, 2000] A. Ruf y R. Horaud. Vision-based guidance and control of robots in projective space. *Proceedings of the 6th European Conference on Computer Vision, Vol. 2, Dublin (Ireland)*, 2000.

[Shapiro y Stockman, 2000] L. Shapiro y G. Stockman. *Computer Vision.* Prentice Hall, New Jersey (USA), 2000.

[Starring, 2004] Moritz Starring. *Computer vision and human skin colour.* PhD thesis, Aalborg University, 2004.

[Trucco y Verry, 1998] E. Trucco y A. Verry. *Introductory Techniques for 3D Computer Vision.* Prentice Hall, New Jersey (USA), 1998.

[Vicente, 2002] J. Vicente. *Métodos visuales de reconstrucción 3D en robots autónomos.* PhD thesis, Universidad Politécnica de Madrid, 2002.

[Zhang y Faugeras, 1992] Z. Zhang y O. Faugeras. A 3d world model builder with a mobile robot. *International Journal of Robotic Research*, 1992.