UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Navegación y Localización de un Robot Móvil, en un entorno cerrado empleando Landmarks Artificiales"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

David Esparza Bórquez

DIRECTOR DE TESIS: "JESUS SAVAGE CARMONA"

México, D.F. **2007.**

Agradecimientos

Deseo agradecer primeramente a mi Tutor el Dr. Jesús Savage por haberme invitado a participar en el grupo de Investigación del laboratorio de Bio Robótica. Fue verdaderamente un placer haber formado parte de este estimulante ambiente de investigación.

Extiendo mis agradecimientos a los profesores del posgrado en Ciencia e Ingeniería de la Computación de la UNAM, en especial al Dr. Boris Escalante, por haberme dado la oportunidad de estudiar en esta gran Universidad.

Agradezco también a Lourdes, Diana, Amalia, Juanita, por su gran ayuda en los aspectos Administrativos y por su siempre actitud servicial.

Gracias a mis colegas universitarios, especialmente a Gerardo Carrera, David Cortes, Carlos Spindola, Carlos Rossel, Javier Rodríguez por su ayuda en los proyectos propios y de equipo, por sus puntos de vista y por compartir muchas interesantes conversaciones.

A Dios, le agradezco por cada día lleno de vida.

A Sofia, el agua que nunca faltó en esta *travesía por el desierto*.

A Paola por compartir conmigo este sueño de ser alumno de la UNAM.

Siempre estaré profundamente agradecido con mi familia y mis más cercanos amigos que siempre están ahí para mí.

Al Conacyt, por su apoyo financiero.

Índice general

Resumen

Esta tesis describe el diseño e implementación de un robot móvil capaz de navegar utilizando información sensorial de tipo visual y de proximidad. El comportamiento esperado es la navegación en tiempo real basado en identificar e interpretar marcas artificiales dispuestas en el entorno, apoyándose en comportamientos reactivos a estímulos inmediatos.

Para los niveles de percepción se proponen las siguientes soluciones: procesar la información recibida de sensores láser mediante la utilización de la técnica de *campos potenciales*, que habilita al robot para alcanzar objetivos y evadir obstáculos. Para el reconocimiento de marcas artificiales se utiliza ARToolKit que comprende librerías de visión por computadora para realidad aumentada. ARToolkit se modificó para obtener la posición de marcas visuales artificiales dispuestas estratégicamente en el entorno, con el único fin de habilitar la navegación de un robot móvil.

El nivel de planeación de caminos propuesto se desarrolla mediante la aplicación del algoritmo de Dijkstra, donde las marcas artificiales corresponden con los nodos que intervienen en el proceso de búsqueda de la solución de Dijkstra. Para ello se contempla la implementación de máquinas de estados que apoyándose en la información sensorial aportada por el láser y la visión, habilite el seguimiento del camino obtenido por Dijkstra. La principal característica de este mecanismo es la interacción constante entre los procesos perceptivos guiados por los estímulos sensoriales.

Capítulo 1

Introducción

1.1. Motivación y Alcance.

La robótica móvil se entiende como la disciplina que investiga desarrollar sistemas capaces de navegar autónomamente en el entorno en el que se encuentran. Para lograrlo, un robot móvil deben poseer habilidades suficientes que le permitan desplazarse libremente entre un punto de partida y un punto de destino (meta), evitando colisionar con áquellos objetos u obstáculos imprevistos en la representación del medio ambiente.

El problema de la navegación se puede englobar en tres tareas principales: la construcción de mapas, la localización y finalmente la planificación de caminos.

La planificación como su nombre lo indica, contempla la capacidad del robot de planificar sus propios movimientos, lo que implica decidir automáticamente el orden de los movimientos a realizar para lograr la tarea especificada. Dada la posición y orientación de inicio y la posición y orientación de la meta a alcanzar, la tarea consiste en obtener un camino constituido por una secuencia de posiciones y orientaciones del robot que le permitan desplazarse sin colisionar con los objetos presentes en el entorno, iniciando en la posición inicial y finalizando en la posición de meta [10]. En caso de encontrar obstáculos inesperados no detectados en el mapa, el robot debe poseer la capacidad de evitarlos.

La evasión de obstáculos es una de las capacidades que debe poseer el robot, ya que el planeador no contempla la presencia de objetos móviles que interfierán durante el seguimiento del camino. A mediados de los 80 aparece la filosofia de robots basados en comportamientos [1], cuya finalidad es dar al robot la capacidad de reaccionar ante los estímulos percibidos por los sensores de a bordo. Con esta filosofia no solo fue posible resolver el problema de evitar obstáculos imprevistos, sino tambien hizo posible la navegación prescindiendo de un mapa del entorno, por ejemplo, mediante sensar y seguir el borde de las paredes de un entorno interior.

La posición del robot respecto a un sistema de referencia es uno de los parámetros mas importantes de los que debe disponer un robot móvil. La odometria consiste en técnicas de posicionamiento que tratan el problema de obtener la posición real a la que se encuentra un

vehículo móvil mediante las lecturas de los sensores internos del robot que contabilizan la distancia recorrida por cada una de sus ruedas. Sin embargo el problema principal al que se enfrenta utilizar este tipo de técnicas, es el deslizamiento indebido de las ruedas, que origina errores en las lecturas de los sensores. Se han propuesto diversas técnicas para corregir este tipo de errores, las cuales no resultan confiables para áreas de navegación extensas. La estimación de la posición del robot puede mejorarse si se incorporan otro tipo de sensores tales como ultrasonidos, sensores láser o camaras de visión. La utilización de marcas artificiales en la navegación de robots ha sido ampliamente utilizada frente a otro tipo de sistemas de navegación que emplean únicamente odometría. Una marca artificial tiene distintas caracteristicas que un robot puede reconocer desde sus sensores de entrada. Puede ser una figura geométrica como un rectangulo, lineas, circulos, etc. En general, una marca artificial tiene una posición fija conocida, relativa a la cual puede localizarse el robot. Antes de que el robot utilice las marcas para realizar la localización, las características de la landmark deben ser conocidas y guardadas en la memoria del robot.

En una revisión del estado del arte acerca de la utilización de marcas artificiales para navegación, es posible observar su aplicación en el seguimiento de caminos de diferentes maneras. Un ejemplo es la utilización de una marca en forma de una línea contínua pintada sobre una superficie clara de tal forma al robot pueda detectarla y seguirla para llegar a los lugares que esten dentro del alcance de la marca. Otro tipo de marca utilizado en [15] [8] es un código de barras colocado en las paredes de un pasillo, que permite a un robot localizarse mientras se desplaza por el pasillo. El problema con este tipo de patrón es el alcance limitado de los sensores de infrarrojo empleados para leer el código de barras.

Existen dos esquemas para la representación del entorno: el esquema métrico y el topológico. En el primer esquema el ser humano puede construir un mapa métrico del entorno de navegación del robot de manera precisa. En el esquema topológico el entorno es codificado mediante una gráfica cuyos nodos representen los lugares del entorno por los que es posible que el robot se desplace libremente, y el espacio libre que existe entre esos dichos lugares es representado por sus arcos. La ventaja de los mapas topológicos es lo compacto de su representación debido a que solo son codificados los lugares distintivos del entorno de navegación, además de que existen varios algoritmos de planeación de caminos desarrollados dentro del campo de la *inteligencia artificial* (*p.ej. Dijkstra, A*, Best First Search, etc.*).

Hay distintos enfoques: en [17] el mapa topológico se obtiene a través de la división del entorno en regiones que corresponden con las celdas de un mapa de rejilla (*grid map*). En [2] los nodos representan lugares distinguidos mediante los sensores internos y los arcos representan los caminos entre estos lugares a los que se asocian unas estrategias de control específicas. En relación con la navegación reactiva, es posible encontrar enfoques topológicos orientados a la navegación basada en comportamientos. En [11] los nodos representan lugares distintivos asociados a comportamientos o a trabajos.

Una parte importante de los algoritmos de localización se basa en la utilización de un mapa del entorno dado con anticipación. Entonces el problema de localización se puede abordar desde dos perspectivas: la localización continua o seguimiento de la posición del robot, y la localización global. En la localización contínua se requiere tener el conocimiento de la posición

inicial del robot, y apartir de la cual es posble estimar la posición en cada movimiento del robot por el entorno. En cambio, en la localización global se desconoce la posición inicial del robot en el mapa, y entonces se procede a estimar la posición del robot en el mapa a partir solamente de la información sensorial y de los movimientos que realice el robot.

El presente trabajo realiza aportaciones útiles para la elaboración de un plan de navegación basándose en información visual procedente de marcas colocadas estratégicamente en el entorno y cuya relación espacial se tiene representada dentro de un mapa topológico. Estas marcas además permiten al robot identificar y alcanzar cada una de las regiones que conforman el mapa topológico, ya que aportan información de su posición respecto al marco local de referencia del robot. Se plantea resolver el problema de la planificación de caminos mediante la utilización del algoritmo de Dijkstra, que aplicado directamente al mapa topológico, permite obtener el camino de menor distancia entre dos lugares distintos. Para resolver el problema de navegación se utilizan técnicas de control reactivo de bajo nivel [1] entre las que destacan el método de los campos potenciales extensamente citado en la literatura [10], y que además resuelve la evasión de obstáculos inesperados.

1.2. Objetivos

El objetivo global que se persigue en esta tesis, es la ***navegación de un robot móvil por un entorno interior conocido empleando landmarks artificiales y evitando colisionar con obstáculos***. Este objetivo se puede dividir en cuatro etapas o sub objetivos. Estos son la construcción del mapa topológico, la planificación del camino dado un mapa topológico del entorno, la ejecución del plan para que el robot alcance la meta propuesta, y, la evasión de obstáculos en línea.

A continuación se detallan cada uno de los tres sub objetivos indicados anteriormente:

Creación del mapa topológico: inicialmente disponemos de un mapa métrico del entorno, en el cual se contemplan obstáculos fijos, como paredes, muebles, puertas, etc. Este tipo de mapa puede ser construido por el diseñador midiendo directamente las posiciones de los objetos en el entorno, o puede haber sido generado en una fase previa en la que se pone al robot a navegar por el entorno sin llegar a colisionar con los objetos presentes y almacenando por medio de sus sensores, la posición de los obstáculos fijos.

A partir del mapa métrico se obtiene una representación topológica, empleando un método de agrupación con características similares a las existentes en los diagramas de Voronoi, conocido como *Cuantización Vectorial*. Este método permite obtener una representación topológica del entorno, en la que se incluye información cartesiana de los nodos que lo conforman. De este mapa se dispone a priori.

Un segundo mapa topológico esta conformado por *landmarks visuales* distribuidas estratégicamente en el entorno de navegación del robot, con la finalidad de hacer más preciso el posicionamiento del robot. Este mapa es construido por el diseñador, en el cual se incluye

información de la posición de los landmarks en el entorno.

Planificación del camino: el siguiente sub objetivo es utilizar el mapa topológico de landmarks artificiales dispuestas en el entorno para encontrar una ruta entre el robot y el destino especificado en el mapa. El destino lo debe indicar el usuario mediante la interfaz de control. Si el destino seleccionado no es accesible por el robot, entonces no se podrá generar el camino hasta éste, en cambio si el destino es accesible para el robot entonces será posible trazar un camino en el mapa topológico contemplando no colisionar con algún obstáculo presente en el entorno.

Planificación del movimiento del robot: empleando como entrada la ruta generada por el planificador, este sub objetivo habilita los comportamientos reactivos de navegación mediante campos potenciales para guiar al robot en el seguimiento de las coordenadas que marcan cada uno de los nodos correspondientes a la ruta del camino. Tras finalizar el recorrido, el robot debe esperar una nueva meta para volver a calcular la distancia hacia ella e iniciar un nuevo desplazamiento.

El método de los campos potenciales requiere conocer la distancia euclídea entre el robot y los obstáculos alrededor, así como la distancia entre el robot y los objetivos a alcanzar. Estas lecturas de proximidad sobre el medio ambiente los obtiene de sus sensores de a bordo, específicamente el sensor láser y el sensor de visión.

Para un mejor entendimiento de los problemas a resolver, y para definir la dirección de esta tesis, es útil presentar la jerarquía de control de nuestro robot autónomo, como se lo puede ver en la figura 1.1.

En el nivel superior de control se ubica el algoritmo para resolver el problema de planificación de rutas. La entrada que requiere es un objetivo marcado por el ser humano especificando las tareas y objetivos que deberá resolver el robot. En este nivel suelen desarrollarse los métodos que resuelven problemas de Inteligencia Artificial (IA). En el nivel intermedio están codificados los procedimientos para resolver las tareas de Navegación mediante Landmarks y evasión de obstáculos. En este nivel existe una interacción continua con el sistema sensorial. Finalmente en el nivel de control de bajo nivel se realiza la labor de control de bajo nivel.

1.3. Organización de la tesis

La tesis se ha organizado en una serie de capítulos que se resumen a continuación:

En el capítulo 2 se presentan los antecedentes, el algoritmo de planificación, y el algoritmo de construcción del mapa topológico. Se muestra un breve ejemplo de la estrategia de caminos mínimos de Dijkstra.

En el capítulo 3 se detalla el método de los campos potenciales que se ha utilizado para la evasión de obstáculos en línea.

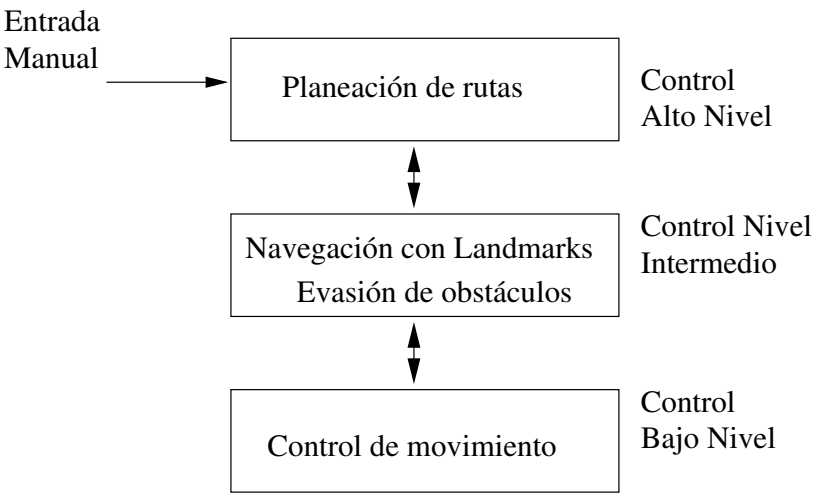


Figura 1.1: Jerarquía de control para robots autónomos. [6]

En el capítulo 4 se muestra como se ha integrado en la arquitectura del robot un conjunto de máquinas de estados para lograr la navegación empleando marcas artificiales, así como información de proximidad para la evasión de obstáculos.

En el capítulo 5 se muestra el software que ha sido necesario desarrollar para la utilización del robot TParacho. Se presenta la implementación del sistema de planeación y navegación.

En el capítulo 6 se discuten los resultados obtenidos.

En el capítulo 7 se exponen las conclusiones y líneas futuras.

En el apéndice A se presenta la plataforma de hardware y de software sobre la que esta constituido esta tesis.

Finalmente se referencia la bibliografía utilizada.

Capítulo 2

Planificación de Caminos

La planificación de caminos resuelve el siguiente problema:

Si el robot está en una posición y debe desplazarse a otra posición distinta, es necesario encontrar un camino a seguir, y sea de preferencia el camino más corto, evitando chocar con los obstáculos presentes en el entorno. Este escenario se ilustra en la figura 2.1.

Existen dos casos:

- Conocemos el entorno de navegación.
- Desconocemos el entorno de navegación y los obstáculos pueden cambiar de posición.

En este capítulo se aborda el primer caso en el cual se tiene disponible un mapa topológico del entorno de navegación construido a partir de marcas artificiales. La planeación de caminos y navegación de un robot móvil puede entoces llevarse a cabo mediante un sistema de visión que facilite identificar lugares mediante el reconocimiento de marcas artificiales colocadas estratégicamente en el entorno de navegación. Estas marcas además permiten al robot identificar y alcanzar cada una de las regiones que conforman el mapa topológico, ya que aportan información de su posición respecto al marco local de referencia del robot.

En la sección 2.1 se presenta la estrategia de navegación. En la sección 2.2 se trata la orno de navegación. En la sección 2.3 se presenta una breve introducción para la construcción del mapa topológico mediante clasificación por cuantización vectorial. Finalmente en la sección 2.5 se expone la ejecución del plan de navegación.

El entorno puede modelarse a partir de un mapa discreto del entorno, com lo

2.1. Estrategia de navegación

Existen dos componentes básicos para lograr la navegación: *planificación de caminos* y *seguimiento de caminos*. La planificación de caminos resuelve la tarea de encontrar una trayectoria que permita al robot alcanzar una posición dentro del entorno. Para ello se requiere en

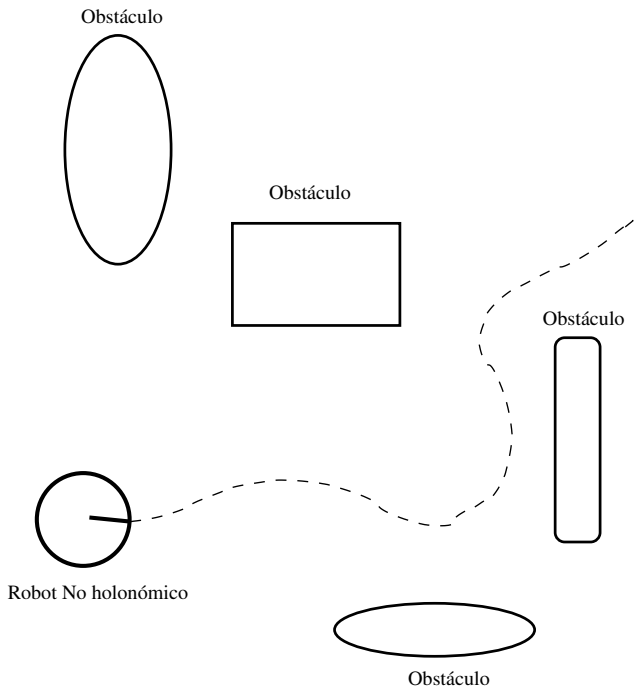


Figura 2.1: El problema básico de planeación y control.

primer lugar, contar con una representación abstracta del entorno contenida en una estructura de datos a la que se le conoce como *mapa del entorno*. Existen diversas representaciones del mapa, pero se dividen principalmente en: métricas y topológicas. Por su parte el seguimiento de rutas guía al robot hacia una posición dentro del entorno, hasta alcanzar el destino especificado por el planeador.

Como se muestra en el pseudo-código de la figura 2.2, el primer paso en el ciclo principal es extraer del mapa la ruta que conecta al nodo inicial con el nodo meta. En este paso el planeador es responsable de obtener la ruta ideal. Una vez que se ha planteado alcanzar una sub meta, el planeador debe envíar los comandos de control a los actuadores del robot para que lo muevan hacia dicha sub meta y, una vez alcanzada se detenga, entonces el robot debe explorar el entorno en busca de marcas visuales y al mismo tiempo localizar su posición en el mapa utilizando la información proporcionada por la marca visual. Mientras el robot esté en movimiento debe utilizar sus sensores para detectar obstáculos desconocidos, y de ser así el robot necesitará desviarse y hacer una nueva planeación de caminos empleando los landmarks dispuestos a su alrededor. Estos pasos serán repetidos hasta que el robot alcance la meta. Si se desconoce la posición inicial del robot, entonces se necesitara de un *localizador global* para determinar dicha posición; ello va más allá de los objetivos de ésta tesis.

En las siguientes secciones se presenta el algoritmo de Dijkstra para encontrar la ruta de menor coste entre dos nodos del mapa topológico (línea 3 en el pseudo-código). Posteriormente se presenta el control de seguimiento de rutas.

1. Planeador (inicio, meta, mapa-topológico, landmarks)
2. while *meta* no alcanzada
3. extraer la ruta más corta desde el *inicio* hasta la *meta*
4. guiar al robot hacia la *sub meta* y parar
5. escanear y localizar utilizando *landmarks*
6. fijar la configuración actual del robot como la posición de *inicio*
7. end while

Figura 2.2: Pseudo-código para el planeador global.

2.2. Representación del entorno de navegación

Para ser de utilidad, la representación del entorno debe tener una relación directa con el mundo externo. La capacidad de un robot de memorizar un determinado entorno para posteriormente llevar a cabo acciones, se basa en representar el entorno sobre una estructura de datos que sea procesable. Si la aplicación es en entornos no explorados, esta estructura debe tener la capacidad de actualizarse fácilmente conforme el robot avance por el camino y deberá facilitar el procesado cuando se pretenda realizar una acción compleja como la re-planificación de rutas o el proceso de auto localización.

Existen diversos métodos para representar el medio ambiente en el espacio de navegación autónoma, destacando los paradigmas *geométrico* y *topológico*. En el paradigma geométrico se obtiene una representación exacta en la que los obstáculos y las zonas libres se modelan de acuerdo a relaciones geométricas absolutas. Esta representación es conveniente cuando el robot necesita conocer con precisión su ubicación en términos de coordenadas métricas. Uno de los métodos más utilizados consiste en dividir el entorno en celdas cuadradas de tamaño fijo a las que se les asigna una determinada probabilidad de ocupación [7]. La desventaja de estos sistemas es que para obtener una representación casi exacta, se deben definir celdas de tamaño pequeño, lo que resulta en un número elevado de celdas y por lo tanto de datos, que implica un mayor tiempo de procesamiento. La ventaja es que se tiene una representación fiable del entorno explorado.

La representación topológica consiste en dividir el entorno en regiones por las cuales el robot pueda deplazarse libremente durante el proceso de navegación. El entorno es representado mediante una gráfica cuyos nodos identifican las distintas regiones en las que se ha dividido el entorno, y los arcos representan las relaciones geométricas (p.ej. la distancia euclideana del espacio libre) que existe entre dichas regiones. La ventaja de este tipo de representación es que produce un menor volumen de información por lo que el procesado resulta computacionalmente menos costoso.

Por un lado, el paradigma geométrico refleja directamente la información sensorial cap-

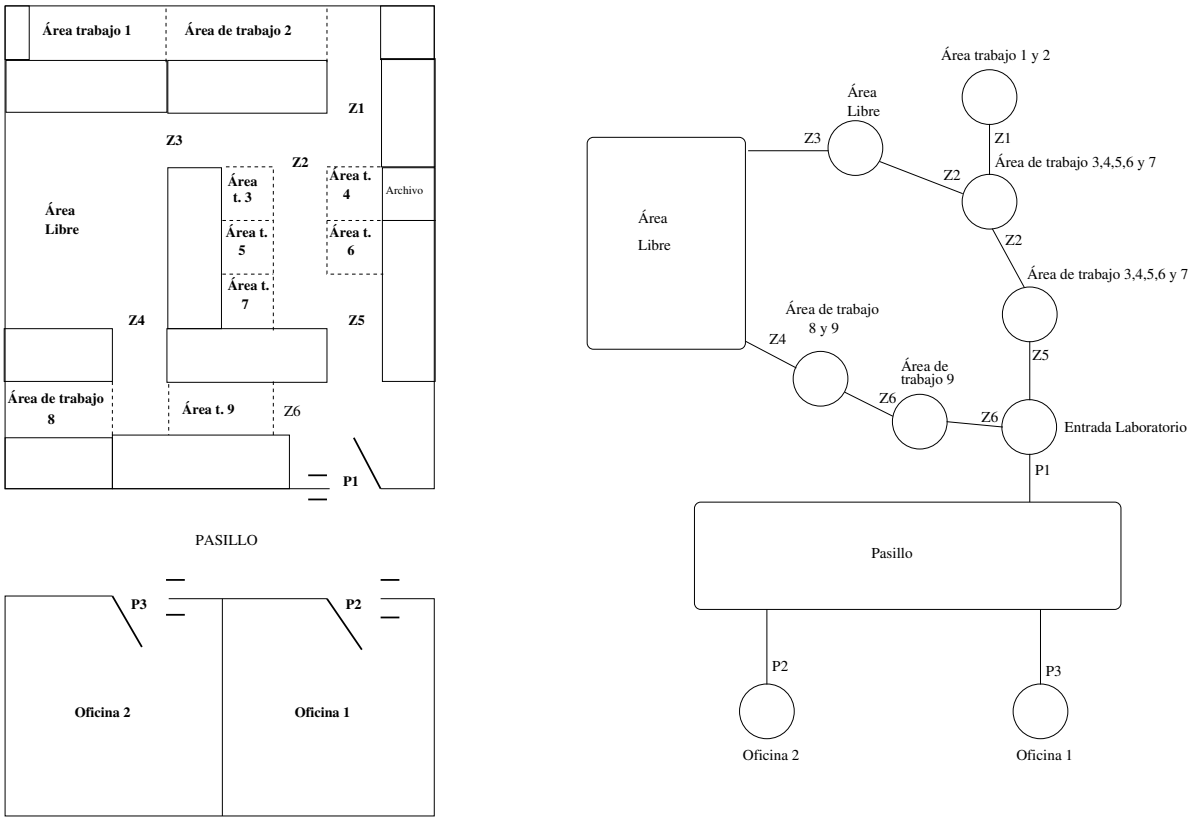


Figura 2.3: Laboratorio de Bio robótica: mapas geométrico y topológico.

tada por el robot y facilita el aprendizaje del entorno ya que basta con tener un mapa donde reflejar las lecturas de los sensores en función de estrategias de actualización que no son costosas, pero tiene la desventaja de introducir errores odométricos debido al problema de deslizamiento de las ruedas [8]. Sin embargo estos errores pueden estimarse y corregirse empleando técnicas de autocalibración [3] y de autolocalización eficientes. Por otro lado en el paradigma topológico se consigue una representación compacta del entorno, y de complejidad directamente proporcional a la del mismo. El problema consiste en obtener de la información sensorial, los nodos y las aristas que integran dicha representación. La tendencia actual es tratar de combinar ambos paradigmas.

2.3. Construcción del Mapa Topológico

Una de las primeras tareas para la navegación de robots móviles, es la construcción de un mapa del entorno. Esta tarea consiste básicamente en obtener un modelo espacial del entorno que permita la navegación del robot móvil.

Los algoritmos para construir mapas topológicos se basan en dos aproximaciones: una consiste en reconocer la estructura topológica directamente, y la otra se basa en construir un

mapa topológico a partir de un mapa métrico. Un ejemplo de la segunda aproximación, lo expone Thrun en [16], en el cual emplea una estructura de celdas de ocupación para construir el mapa métrico, y a partir de éste extraer el mapa topológico.

Un método clásico para la construcción de mapas topológicos es el diagrama de Voronoi, que permite encontrar, por llamarlo de un modo, el esqueleto del entorno, que se corresponde con los puntos del espacio libre, equidistantes a todos los obstáculos alrededor.

2.3.1. Diagramas de Voronoi

El diagrama de Voronoi (GVD), básicamente consiste en la proyección del espacio libre, en una estructura de curvas unidimensionales proyectadas sobre dicho espacio libre.

El diagrama de Voronoi tiene la propiedad de ser una deformación de retracción [4], que garantiza la continuidad del camino en el GVD. Una deformación de retracción es la imagen de la función continua de retracción RM, tal que:

$RM(q) = q$, para todo q en el GVD
$RM(q) = q'$ para algún $q \in Q_{libre}$ y $q' \in$ GVD.

En otras palabras, si q esta en el GVD, entonces la imagen de q es q, y en este caso $RM(q) = q$. Si q esta dentro del espacio libre pero no dentro del GVD, entonces la imagen de q es la q dentro del GVD que se obtiene retirándose de aquel punto perteneciente al obstáculo más cercano hasta alcanzar el GVD, y entonces $RM(q) = q'$.

Se puede notar que la conectividad del GVD es una consecuencia de la continuidad de la función RM. Por lo tanto existe un camino que conecta q_{inicio} y q_{meta} si y solo si existe un camino en el GVD que conecte q'_{origen} con $q_{destino}$, donde $q'_{inicio} = RM(q_{inicio}$ y $q'_{meta} = RM(q_{meta})$.Este algoritmo agrupa aquellos puntos que están separados a la misma distancia respecto de un par de obstáculos.

$$F_i = \{q \in Q_{libre} \quad | \quad i(q) \leq d_h(q) \quad \forall h \neq i\} \tag{2.1}$$

donde $d_i(q)$ es la distancia a un obstáculo QO_i de q, y en este caso, $d_i(q) = min_{C \in QO_i}$, $d(q, c)$.

Función de Retracción

Supongamos que se tiene un conjunto Q_{libre} que consta de las posiciones libres de obstáculos dentro del entorno, entonces la función de retracción RT, construye un subconjunto Q_v continuo de Q_{libre}.

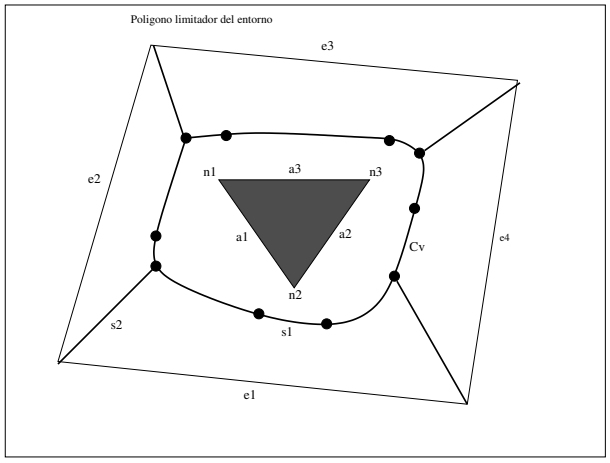


Figura 2.4: Retracción del espacio libre en un diagrama de Voronoi.

Lo que se busca, es la existencia un camino desde una configuración inicial q_{inicio} hasta otra final q_{meta}, supuestas libres de obstáculos, si y solo si existe una curva continua desde $RT(q_{inicio})$ hasta $RT(q_{meta})$. Es por ello que el GVD resulta del lugar geométrico de las configuraciones que se encuentran a igual distancia de los dos obstáculos más próximos del entorno, y esta formado por dos tipos de segmentos: rectilíneos y parabólicos. El tipo de segmento corresponde con la forma de los obstáculos más cercanos que se encuentren de frente uno al oro. De esta forma, el lugar geométrico de las configuraciones que se hallan a igual distancia de dos aristas hacia dos obstáculos diferentes, es una línea recta, mientras que en el caso de un vértice y una arista, resulta en una parábola.

Por ejemplo, si el conjunto Q_{libre} define las posiciones libres en el entorno, la función de retracción RM construye un subconjunto Q_v continuo de Q_{libre}:

$$RM(q) : Q_{libre} \rightarrow Q_v / Q_v \subset Q_{libre} \tag{2.2}$$

Se dice entonces que existe un camino que une la posición inicial q_{inicio} a la configuración final q_{meta} ambas libres de obstáculos, si existe una curva continua entre $RM(q_{inicio})$ y $RM(q_{meta})$.

En la figura 2.4 se muestra el entorno ocupado por un polígono de aristas (e1,e2,e3 y e4), y un obstáculo triangular. El diagrama de Voronoi corresponde a las líneas de trazo grueso que representan la retracción del espacio libre en una red continua de curvas. El segmento $s1$ es el lugar geométrico de los puntos equidistantes entre la arista $e1$ y el vértice $n2$. De igual manera, el segmento rectilíneo s_2 cumple la condición anterior, pero con respecto a las aristas $e1$ y $e2$.

2.4. Creación del mapa

Se pueden usar representaciones del medio ambiente utilizando mallas de celdas espaciadas uniformemente. Cada celda de la malla contiene un valor que indica la presencia o ausencia de un obstáculo en la región correspondiente del ambiente.

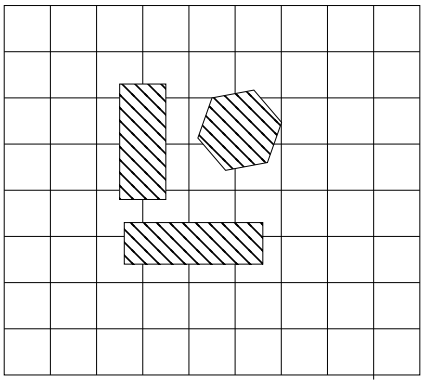


Figura 2.5: Mapa basado en mallas.

Los objetos presentes en el medio ambiente son modelados por polígonos. Los polígonos están restringidos a que las esquinas de un polígono no crucen a otro polígono.

Descripción de áreas poligonales

Una de las formas más sencillas para representar a un polígono es un lista ordenada de sus vértices. Pares sucesivos de vértices definen esquinas sucesivas de el polígono imponiendo un ordenamiento consistente de los vértices, ellos también definen lo que está afuera o dentro del polígono.

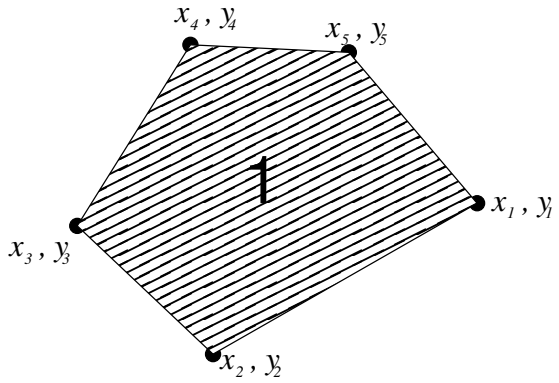


Figura 2.6: Un polígono puede definirse a partir de pares sucesivos de vértices.

$$Poligono1 = ((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)) \tag{2.3}$$

La ecuación general de una línea o segmento en forma paramétrica es:

$$x = x_0 + mt$$
$$y = y_0 + mt$$

donde z es el parámetro y (x_0, y_0) es el punto de la recta para $t = 0$. Adoptando la convención que t va de 0 a una orilla (llamada K) y $t = 1$ a la otra orilla (llamada L). Entonces se puede escribir:

$$x = X_K + (X_L - X_K)t$$
$$y = Y_K + (Y_L - Y_K)t$$

Esta forma paramétrica puede ser convertida en forma implícita para dar una ecuación sencilla para una línea que contiene los puntos K y L:

$$(Y_K - Y_L)x + (X_L - X_K)y + (X_K \circ Y_L - X_L \circ Y_K) = 0 \tag{2.4}$$

Entonces puntos que están de un lado de la línea satisfacen la desigualdad:

$$(Y_K - Y_L)x + (X_L - X_K)y + (X_K \circ Y_L - X_L \circ Y_K) < 0 \tag{2.5}$$

mientras que los puntos del otro lado satisfacen:

$$(Y_K - Y_L)x + (X_L - X_K)y + (X_K \circ Y_L - X_L \circ Y_K) > 0 \tag{2.6}$$

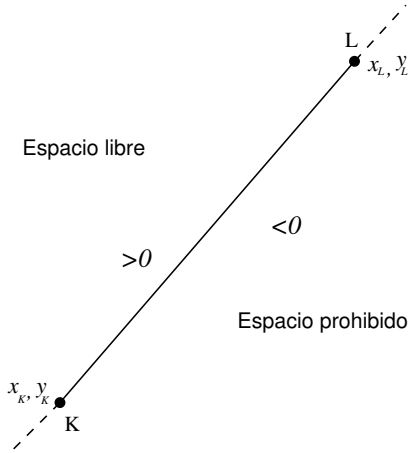


Figura 2.7: Espacio prohibido y espacio libre en un segmento de polígono.

2.5. Planificación Topológica

La planificación topológica es una manera *natural* de planificar el movimiento del robot. "Sigue recto hasta la marca 1, gira a la izquierda y continúa por el pasillo hasta que encuentres la marca", es un tipo de planificación topológica que esta estrechamente relacionado con el concepto de punto distinguible o *landmark*. Nuestro robot debe ser capaz de identificar estos landmarks [1] normalmente mediante sensores. Un caso especial de landmarks son los landmarks artificiales que trataremos en el capítulo 5.

2.5.1. Algoritmo de Dijkstra

Existen distintas operaciones que pueden aplicarse sobre gráficas. Por ejemplo, los algoritmos de Kruskal y de Prim, encuentran la manera más eficiente de atravesar completamente una gráfica. No obstante, si se requiere calcular la distancia entre dos vértices determinados, existe un método alternativo: el algoritmo de Dijkstra. Este algoritmo determina la distancia (costo) entre un vértice con todos los demás vértices pertenecientes a una gráfica conexa. Su nombre pertenece a su desarrollador el holandés Edsger W. Dijkstra.
El Algoritmo de Dijkstra es aplicado directamente al problema de planificación, ya que permite encontrar la ruta de menor coste a partir de un mapa topológico (codificado en un grafo) del entorno de navegación. El funcionamiento de este algoritmo, consiste ir explorando cada uno de los caminos que parten del vértice origen y que lleva a todos los demás vértices. El algoritmo se detiene cuando ha encontrado el camino de menor coste, para todos los nodos del mapa topológico conexos desde el nodo origen [5].

El algoritmo de Dijkstra tiene como premisa que la gráfica sobre la que se realizará la búsqueda sea conexa, es decir, que desde cada nodo se pueda alcanzar al resto de los nodos a través de las aristas que los unen, o dicho de otro modo, no deben existir islas de nodos desconectados del resto de la gráfica.

Este algoritmo es de complejidad $O(n^2)$, donde n representa el número de vértices, ya que realiza la búsqueda desde su posición origen hasta el resto de los nodos.

Implementación

Empleando la técnica de construcción de mapas mediante rejillas de celdas cuadradas descrito en la sección 2.4, se crea una gráfica que especifica la relación entre los distintos nodos y el coste de pasar de uno a otro. Se considera como nodos de *origen* y *destino* las posiciones del robot y de la meta a alcanzar. Enseguida se actualizan las conexiones de todos los nodos del mapa topológico con los nodos de origen y meta.

La estructura principal sobre la que se calcularán los caminos mínimos es un vector cuyas posiciones se corresponderán con cada uno de los nodos del grafo por los que pasará el camino mínimo. A este vector lo denominaremos *vector de caminos mínimos*. Para cada posición del vector, se incluyen los siguientes valores:

[1] Un landmark es una característica que se puede identificar y que es de interés para utilizarla en la navegación ya que nos sirve para reconocer lugares dentro del entorno.

- El costo del camino calculado desde el nodo origen hasta el nodo correspondiente a la posición actual del vector.

- Un indicador de nodo visitado
- El coste del camino calculado, que será un valor real con la suma de las distancias entre los nodos del vector de camino calculado.

La obtención del camino mínimo entre el nodo origen y el nodo destino, consiste en seguir los siguientes pasos iterativamente:

1. DIJKSTRA (V, E, $inicio$, c_f)
2. for (each $v \in V$) $dist[v] \leftarrow \infty$
3. $dist[inicio] \leftarrow 0$
4. $PQ \leftarrow$ Cola-de-Prioridad de V ordenada por $dist$
5. while ($PQ \neq 0$)
6. $u \leftarrow PQ$.desencolar
7. for each $v \in PQ$ adyacente a u
8. $if(dist[v] > (dist[v] + peso(u, v, padre[u]))$
9. $dist[v] \leftarrow dist[v] + peso(u, v, padre[u])$
10. $padre[v] \leftarrow u$
11. end for
12. PQ.reordenación
13. end while

Figura 2.8: Algoritmo de Dijkstra

El algoritmo devuelve un vector de camino mínimo con la ruta de menor coste, partiendo desde el nodo de inicio, hasta cada uno de los nodos conexos a éste.

Ejemplo

Enseguida se muestra un ejemplo del funcionamiento del algoritmo de Dijkstra para un mapa métrico, cuyos costes equivalen a la distancia euclídea entre los vértices. Se transforma

el mapa métrico en un mapa topológico equivalente, como se muestra en la figura 2.9.

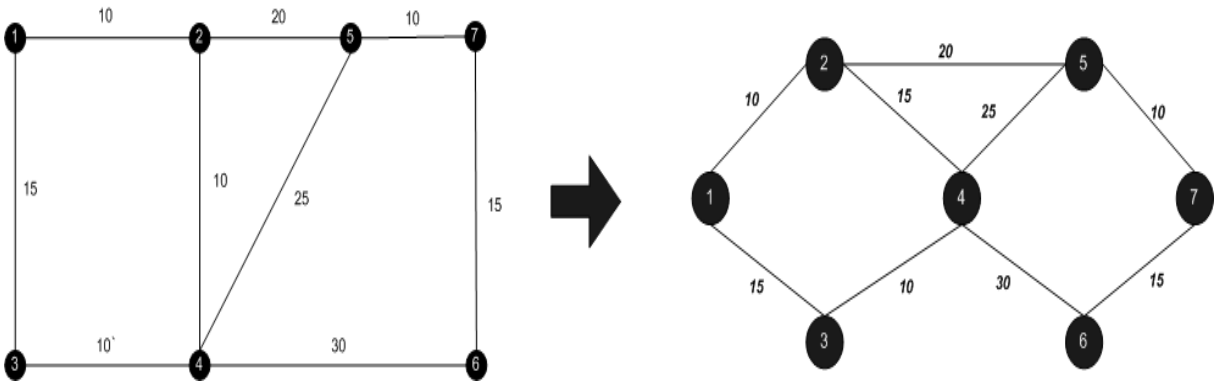


Figura 2.9: Ejemplo del Algoritmo de Dijkstra. gráfica Inicial

Partimos del nodo inicial 1, que tendrá costo 0, y cuyo camino mínimo es el mismo nodo. El resto de los nodos tendrán un costo infinito, además de no tener todavía un camino mínimo asignado, como se muestra en la inicialización de la figura

Calculamos los caminos desde el nodo inicial hasta los dos nodos adyacentes numerados como 2 y 3. Para el nodo 2, el camino mínimo es: 1,2 con coste 10, dado que este valor es menor que el coste inicial del nodo 2 (∞). De forma análoga, para el nodo 3, su camino mínimo tiene un coste de 15. La primer iteración, y el estado de los costes y de los caminos asignados se muestra en la figura 2.10.

Una vez actualizados todos los nodos hijos del nodo de 1, lo marcamos como 'tratado', marcando a VERDADERO el indicador booleano de camino mínimo, y pasamos a explorar el nodo 2, que resultó ser el nodo no tratado de menor coste.

Los nodos adyacentes al nodo 2 son los nodos 4 y 5. Para el nodo 4, se tiene la ruta 1,2,4, de coste 25. De igual forma para el nodo 5 se calcula la ruta con coste de 30, que pasa por los nodos 1,3,5. Una vez actualizados los nodos hijos del nodo 2, lo marcamos como "tratado" marcando a "VERDADERO" el indicador booleano de camino mínimo. La gráfica queda como como se muestra en la figura: 2.10.

En la tercera iteración del ciclo, se selecciona el nodo 3 como nodo actual, por ser el nodo no tratado de menor coste. El nodo adyacente al nodo 3, que no ha sido tratado es el nodo 4. El costo del camino 1,3,4 es de 25, que es igual al coste del camino 1,2,4, ya explorado, por lo que no es necesario actualizar el camino hacia este nodo.

En la cuarta iteración, se parte del nodo 4, cuyos nodos adyacentes no tratados son el 5 y el 6. Recalculando el costo de las rutas en ambos nodos, no se encuentran rutas de menor coste que los costes encontrados previamente, por lo que se marca el nodo 4 como tratado, y pasamos al siguiente ciclo.

Se selecciona el nodo 5 como nodo actual, por ser el nodo no tratado de menor coste asignado. Desde este nodo calculamos una nueva ruta para el nodo 7 que es el único nodo adyacente no tratado. El costo de la ruta 1,2,5,7 asignado a este nodo es de 40.

Marcamos el nodo 5 como tratado y continuamos con el nodo 6 que tiene un coste de 35. El único nodo adyacente a éste y que no ha sido tratado es el nodo 7, cuya ruta 1,2,4,6,7 de coste 70, resulta ser mayor que el coste 40 que previamente tenia asignado en la ruta 1,2,5,7, por lo tanto no es necesario actualizar su coste. La figura 2.11 muestra los caminos generados hasta el momento en la quinta iteración.

Tras marcar el nodo 6 como tratado, solamente falta el nodo 7 por tratar. Esta es una condición de finalización del algoritmo, puesto que ha sido tratado cualquier nodo adyacente al nodo 7. Se muestra la última iteración los costes y caminos mínimos finales.

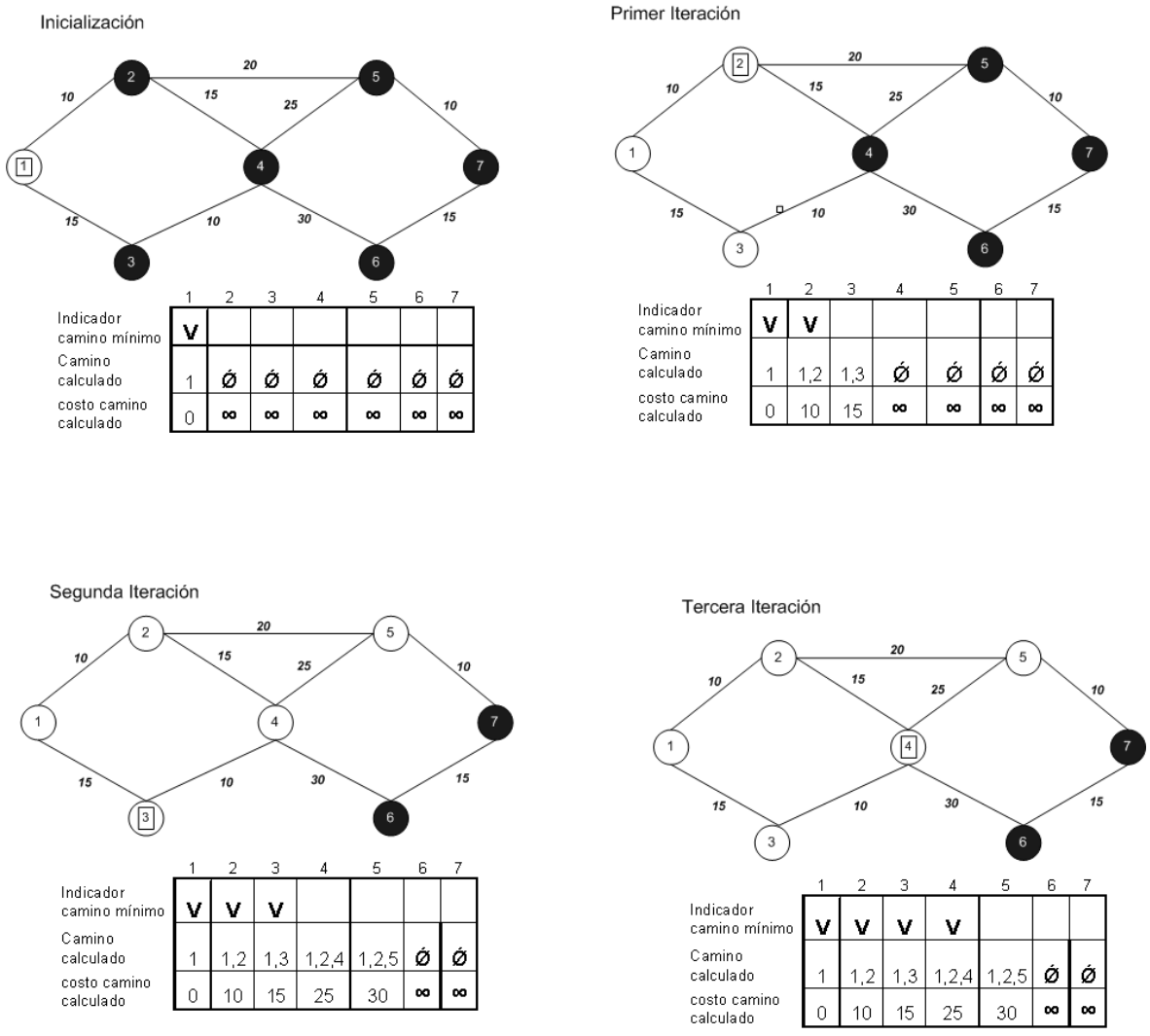
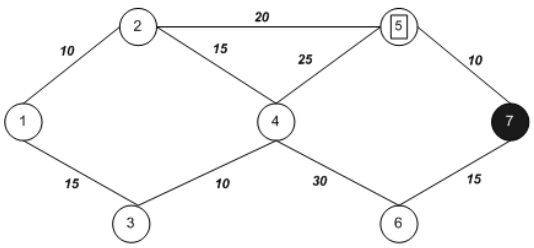


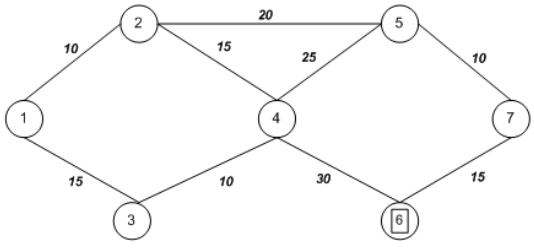
Figura 2.10: Ejemplo del Algoritmo de Dijkstra. Iteración 1 a 3

Cuarta Iteración



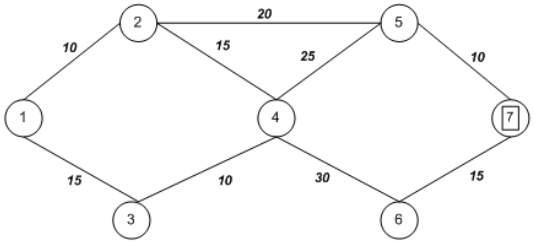
	1	2	3	4	5	6	7
Indicador camino mínimo	V	V	V	V	V		
Camino calculado	1	1,2	1,3	1,2,4	1,2,5	1,2,4,6	Ø
costo camino calculado	0	10	15	25	30	55	∞

Quinta Iteración



	1	2	3	4	5	6	7
Indicador camino mínimo	V	V	V	V	V	V	
Camino calculado	1	1,2	1,3	1,2,4	1,2,5	1,2,4,6	1,2,5,7
costo camino calculado	0	10	15	25	30	55	40

Última Iteración



	1	2	3	4	5	6	7
Indicador camino mínimo	V	V	V	V	V	V	V
Camino calculado	1	1,2	1,3	1,2,4	1,2,5	1,2,4,6	1,2,5,7
costo camino calculado	0	10	15	25	30	55	40

Figura 2.11: Ejemplo del Algoritmo de Dijkstra. Iteración 4 a 6

2.6. Ejecución del plan

Una vez planificada la ruta sobre el mapa topológico, debemos enviar las órdenes correspondientes al robot para recorrer la ruta generada por el algoritmo de Dijkstra, que esta almacenada en un vector de nodos, conocido como *vector de trayectoria*. Este vector se inicializa a cero cada vez que se inicia una etapa de planificación, y se rellena con la ruta entre el robot y la meta, siempre y cuando Dijkstra haya encontrado un camino entre las posiciones de origen y destino.

Una forma de recorrido de la trayectoria consiste en repetir el ciclo de desplazarse para alcanzar cada nodo del vector de trayectoria desde el nodo de inicio hasta el nodo meta como se indica a continuación:

1. Obtener la posición x_d, y_d del destino a alcanzar.

2. Obtener la posición actual del robot x_r, y_r a través del sistema odométrico.

3. Trazar la trayectoria a seguir, que consiste de la magnitud m y dirección θ entre la posición actual y la posición de destino:

   $\theta = \tan^{-1} \frac{(x_d - x_r)}{(y_d - y_r)}$; si $x_r \neq x_d$

   $magnitud = \sqrt{(x_d - x_r)^2 + (y_d - y_r)^2}$

4. Dar el comando de movimiento al robot compuesto por la magnitud y dirección de movimiento obtenidos en el paso anterior.

Capítulo 3

Planificación del movimiento del robot

3.1. Introducción.

La planificación topológica construye un camino ideal desde el punto de vista euclídeo, y para ello cuenta simplemente con información a priori del entorno, es decir, se considera que el terreno de navegación permanece estático, puesto que esta tarea se desarrolla antes de llevar a cabo la ejecución de la tarea de navegación. Ante tal situación, el robot apoyado de las mediciones provenientes de sus sensores externos, puede advertir la presencia de obstáculos inesperados mientras avanza en el seguimiento de la ruta impuesta por el planeador global. Estos datos son utilizados por un módulo de planeación local, el cual tiene la tarea de modificar la ruta original para evitar colisionar con aquellos obstáculos que interfieran en el seguimiento del camino global.

Para conseguir realizar esta tarea, existen dos enfoques:

- Aquellos basados en técnicas de planificación: un método muy extendido es el ajuste de curvas sobre la ruta construida, que conserva las características cinemáticas del camino. Comúnmente este tipo de planificación se realiza a priori.

- Navegación reactiva: los sistemas basados en control reactivo, permite al robot responder en tiempo de ejecución ante el estado actual de entornos dinámicos poco estructurados.

El método propuesto por esta tesis, pertenece al segundo enfoque. En éste se considera que los obstáculos presentes durante la tarea de navegación permanecen estáticos, y que se cuenta con el camino global construido.

Formalización

Sea R un objeto rígido móvil (robot) en un espacio euclidiano C, llamado espacio de configuración, representado como R^2, y tenemos q objetos rígidos estáticos, $B_1, B_2, ..., B_q$ distribuidos en C, que llamaremos obstáculos. Nuestro problema consiste en dada una posición y orientación inicial (s) y final (t) de R en C, tenemos que encontrar un camino P entre ambas posiciones de forma que R no colisione con ningún obstáculo B_i.

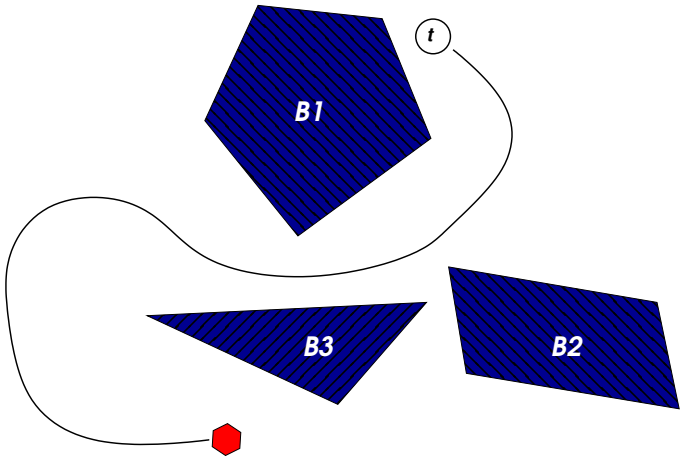


Figura 3.1: Ejemplo.

De esta manera se aborda el planteamiento del problema de evasión de obstáculos (sección 4.2), donde se exponen las condiciones que deben existir para considerar el cambio de trayectoria por la presencia de un obstáculo. Enseguida se expone formalmente el método de los campos potenciales (sección 4.3). Finalmente se exponen las limitaciones para las cuales los campos potenciales pueden caer en un mínimo local (sección 4.4).

3.2. Definición del problema.

La evasión de obstáculos estáticos mientras el robot está en movimiento implica en primer instancia, la detección del obstáculo para posteriormente determinar la acción de control que dirija al robot fuera de su alcance. La detección dependerá de la información suministrada por los sensores externos del robot.

Uno de los enfoques establece un radio de seguridad que permita avanzar al robot sin tener contacto con el obstáculo. Otro enfoque considera un área circular que encierra al obstáculo

evitando así no colisionar con el mismo.

El enfoque reactivo que se aborda en esta tesis considera seleccionar los comportamientos adecuados que lleve al robot fuera del alcance del obstáculo, según la información perceptual que posee del entorno. Una vez en un punto de seguridad, el planeador global apoyado del conocimiento que tiene de la posición del vehículo, hace el replanteamiento de la ruta hacia la meta original.

3.3. Planificación por campos potenciales

Conocido también como *Navegación mediante evasión de obstáculos en línea*; éste método genera un campo vectorial que representa un espacio navegacional sujeto a una función potencial arbitraria diferenciable. El valor de una función potencial puede ser visto como energía y el gradiente del potencial como una fuerza. La teoría de campos potenciales considera al robot como *una partícula positivamente cargada*, influenciada por un campo de potencial (campo de fuerzas). Las fuerzas pueden ser de dos tipos:

- Fuerzas de atracción: Una meta o posición a alcanzar.
- Fuerzas de repulsión: Obstáculos.

La posición meta es una de esas fuerzas de atracción y por ende se le considera *negativamente cargada*. A su vez, los obstáculos poseen una *carga positiva* y forman una fuerza repulsiva direccionando al robot fuera de su alcance. La combinación de fuerzas atractivas y repulsivas dirigen cuidadosamente al robot de una localidad inicial hacia una localidad meta, al mismo tiempo que permiten al robot ir evitando obstáculos.

La planificación mediante campos de potencial, se basa en llevar a cabo la siguiente secuencia de acciones:

- Calcular el potencial U(p) que actúa sobre el robot en la posición actual y según la información recabada de los sensores.
- Determinar el vector de fuerza artificial F(p) según la expresión (3.5)
- En virtud del vector calculado construir las instrucciones adecuadas para los motores del robot que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por F(p).

La iteración continua del ciclo antes presentado, es una forma de navegación reactiva basada en campos potenciales. El comportamiento del robot está muy ligado al resultado la acción de las fuerzas potenciales de atracción y de repulsión.

3.3.1. Obstáculos y Meta

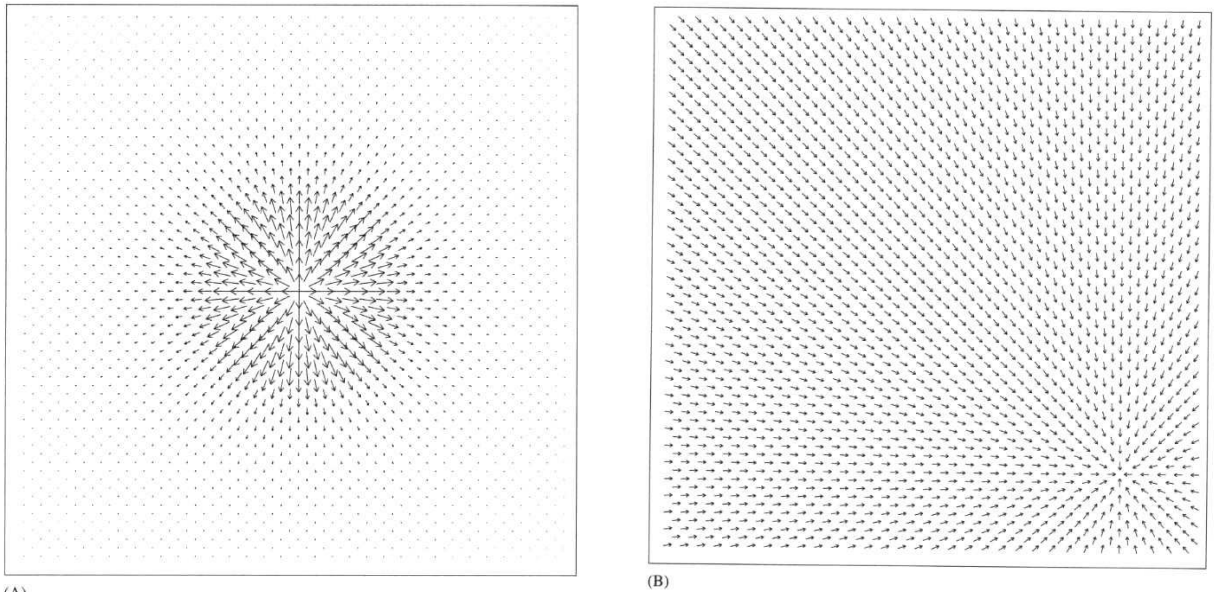
Para desarrollar el método de los Campos Potenciales, seguimos la aproximación usual (como en electrostática) donde la magnitud de la fuerza virtual depende del cuadrado de la distancia entre el robot y el obstáculo. La dirección de la fuerza es en la dirección de la linea más corta que une una posición x, y del entorno con el robot. Si la posición del objeto en cuestión es un obstáculo, la fuerza será de repulsión ya que hipotéticamente se considera que el robot y el obstáculo tienen cargas del mismo signo. La meta o destino tiene un comportamiento opuesto, es decir el robot y la meta se considera que tienen una carga de signo contrario, por lo que la posición de la zona libre en cuestión atraerá al robot. Podemos expresar la fuerza virtual sobre el robot debida a un obstáculo o como:

$$F_{rep} = K\frac{Qq_o}{d^2}\hat{u} \tag{3.1}$$

Donde,

F_{rep}= Fuerza asociada con el obstáculo o
K = Constante de proporcionalidad
Q = "Carga" asociada con el Robot
q_o = "Carga" asociada al obstáculo
$d = |x_r - x_o|$
x_r = Vector de Posición del Robot
x_o = Vector de Posición del obstáculo
$\hat{u}$ = vector unitario en la dirección de o

$$\hat{u} = \frac{x_r - x_o}{|x_r - x_o|} \tag{3.2}$$



(a) (b)

Figura 3.2: Campos potenciales en (a) debido a un obstáculo y en (b) atractivos para una meta situada en la parte inferior derecha de la sub-figura. [1]

Seguiremos la convención de considerar que el robot y los obstáculos están negativamente cargados, mientras la meta está positivamente cargada. Por lo tanto fuerzas resultantes positivas serán repulsivas, mientras que las fuerzas negativas serán atractivas.

Por conveniencia optamos por asignar a 1 el valor de q_o y de Q, ajustando únicamente la magnitud de la fuerza K. La información de los obstáculos es procesada directamente de los sensores láser y de visión, y es representada como puntos en el espacio bidimensional.

Debe tomarse en cuenta la dirección hacia la meta a alcanzar, sin importar el hecho de donde se encuentra el robot en circunstancias normales, para lo cual se fija a un valor constante la magnitud de la fuerza de atracción. Por lo tanto definimos la fuerza de atracción por:

$$F_{atr} = A_t \hat{v} \tag{3.3}$$

Donde,

F_{atr} = Fuerza del vector asociado con el Objetivo t
A_t = Constante de proporcionalidad
x_t = Coordenadas cartesianas del Objetivo t.
$\hat{v}$ = vector unitario en la dirección de t

$$\hat{v} = \frac{x_r - x_t}{|x_r - x_t|} \tag{3.4}$$

La fuerza total virtual sobre el robot es obtenida mediante la adición vectorial de todas las fuerzas ejercidas por los obstáculos y las sub metas. Así tenemos que si $O = \{$Todos los obstáculos percibidos por los sensores$\}$, entonces la fuerza neta sobre el robot F_{net} esta dada por:

$$F_{net} = \sum_{\forall_o \in O} F_{rep} + F_{atr} \tag{3.5}$$

Podemos expresar esto en términos de sus componentes x,y:

$$\theta = \tan^{-1}(\frac{F_y}{F_x}) \tag{3.6}$$

Así obtenemos la dirección del movimiento del robot, que es en la dirección de la fuerza neta (sujeta a ciertas condiciones) y de ahí que la ecuación (3.6) define el ángulo de giro que debe seguir el robot.
(3.6)

3.3.2. Limitaciones de la navegación mediante campos potenciales.

El problema en este tipo de método, recae en la presencia de mínimos locales, o sea, lugares que no indican el verdadero mínimo global, y el potencial resulta nulo. Una situación así lleva al robot a una posición que no es el destino que se busca alcanzar, o pudiera quedar atrapado en una zona cerrada, por ejemplo, avanzando de un lado a otro pero sin salir de dicha área. Esto puede verse claramente en la figura 3.3.

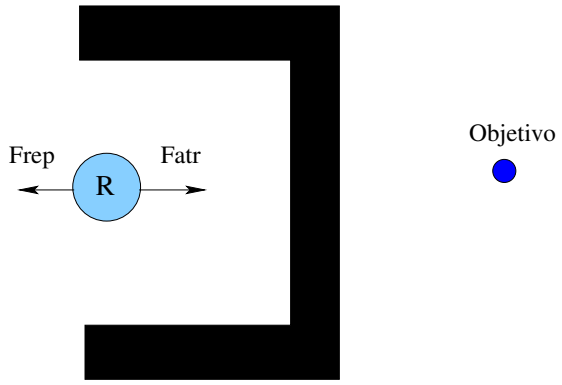


Figura 3.3: En caso de caer en un mínimo local, el robot no se moverá.

3.3.3. Algunas soluciones para enfrentar las limitaciones de los campos potenciales.

Para lograr que la solución al problema de evasión de obstáculos sea completa, puede usarse un planificador global cuando se detecten este tipo de situaciones. El planificador global podría entonces trazar desde la posición actual del robot, un camino diferente hacia el objetivo, sin embargo el coste computacional resulta excesivamente alto para aplicarlo en cada ciclo en un sistema de tiempo real, en mapas del entorno que incluyen una gran cantidad de nodos.

Otra de las alternativas es emplear comportamientos que dirijan al robot hacia cualquier lugar, es decir, den un movimiento aleatorio para lograr sacar al robot del mínimo local.

Una tercer alternativa de solución a este problema, es modificar el algoritmo para evitar caer en dichos mínimos locales, aunque resulta laborioso. También puede estructurarse el entorno de navegación para lograr evitar caer en un mínimo local, por ejemplo utilizando objetos en forma de círculos (Rimon y Koditschek, 1988).

Capítulo 4

Navegación mediante Landmarks Artificiales

4.1. Introducción

La navegación entre dos puntos se plantea como la ejecución de un plan compuesto por una serie de nodos unidos mediante arcos así como de un grupo de comportamientos a ejecutar, siendo los nodos los lugares o sub-metas a alcanzar en el plan. Las habilidades del robot para alcanzar dichos objetivos y evitar obstáculos son fundamentales para conseguir el logro del plan generado.

Para mejorar las habilidades del sistema de navegación topológico que utiliza solamente odometría, es necesario incorporar información visual, que ha sido ampliamente utilizada en el campo de la robótica como sensor en la navegación de robots, específicamente landmarks artificiales.

En este capítulo presentamos la aplicación de las librerías ARToolkit [9] que incorpora información visual a un sistema de navegación basado mapas topológicos, y le proporciona una habilidad adicional consistente en la capacidad de encontrar los nodos que corresponden al mapa topológico del entorno, y así dirigirse hacia ellos de forma precisa. Esta habilidad surge de la interacción de comportamientos que toman sus datos de entrada, de los sensores láser y odométrico, con otros que utilizan solamente la información extraída a partir del sistema de visión. Tanto unos como otros han sido diseñados utilizando el esquema de máquinas de estados (FSA), lo que nos permite una manipulación adecuada de la imprecisión subyaciente en los datos sensoriales.

4.2. Landmark Artificial

Un landmark es una característica que se puede identificar y que es de interés para utilizarla en la navegación (nos identifica lugares).

Puede ser un objeto o lugar:

- Una linea en el piso
- Un dibujo en la pared
- Un cruce de pasillos
- Una puerta

Hay dos clases de landmarks: artificiales y naturales. Los landmarks naturales son mejores cuando son hechos por el hombre, como en el caso de ambientes bien estructurados, de los cuales se busca enfocar en estructuras u objetos: una puerta, el cruce de pasillos, etc. De ahí se toma la definición de que los landmarks naturales son aquellos objetos o características que están presentes en el medio ambiente y entre otras cosas tienen la función de permitir la navegación de un robot; por otro lado, los landmarks artificiales son objetos o marcas especialmente diseñados para ser colocados en el ambiente de navegación con el único fin de facilitar la navegación de un robot. Ejemplos de landmarks artificiales son códigos de barra, un dibujo, líneas en el suelo, puntos de color, etc. [8].

Entre las características que deben poseer los landmarks se tienen:

- Debe ser reconocible: si el robot no es capaz de encontrarlo, no es de utilidad.
- Se reconoce mediante sensores.
- debe soportar la tarea a desarrollar:
- Si la tarea es planificación, simplemente identificando el landmark es suficiente.
- Si la tarea es estimación de la posición, esta estimación nos la debe proporcionar el propio landmark.
- Debe ser perceptible desde distintos puntos de vista.

Las ventajas que poseen los landmarks, es que reducen los errores de odometría, y permiten localizarse dentro de un mapa. Entre los inconvenientes se encuentra que no es sencillo encontrar landmarks en el entorno, además de que es la identificación de un landmark de forma única depende del sistema de percepción.

La idea de utilizar landmarks artificiales para facilitar la navegación en robots móviles, no es nueva, por ejemplo en aplicaciones marítimas y aeronáuticas se utiliza la colocación de emisores activos (beacons) en el medio ambiente para facilitar la navegación. En dichas

técnicas se obtiene la posición midiendo el ángulo de recepción de tres o más emisores, de los cuales debe conocerse su posición global en el entorno. Este tipo de estimación se ha aplicado a los robots móviles para exteriores, y ofrece la ventaja de obtener la localización absoluta del robot en áreas extensas.

En sistemas de visión artificial se emplean landmarks artificiales para lograr el mismo objetivo. La ventaja de utilizar marcas artificiales (pasivas), es que son fáciles de reconocer en el entorno. Normalmente se suelen diseñar con patrones en forma de códigos de barras, y de figuras geométricas para que el reconocimiento en el entorno sea óptimo incluso en condiciones de baja luminosidad.

4.3. Arquitectura Basada en Comportamientos

Muchas de las tareas para las cuales se diseñan robots pueden ser divididas en *comportamientos* para su ejecución secuencial. No obstante, esta secuencia de acciones no es lineal ya que los comportamientos no se ejecutan en un orden fijo, es decir, un comportamiento activo puede ser seguido por otro entre varios posibles, y la activación del comportamiento sucesor depende de la interacción del robot y su entorno.

4.3.1. Comportamiento de seguimiento de rutas.

Este comportamiento realiza la detección y seguimiento del landmark específico que se le indique por parte del planeador, en función de la posición donde se encuentre el robot en un momento dado. Este comportamiento es capaz de detectar el conjunto de landmarks que hay en una imagen concreta captada por la cámara de visión, e informar sobre si un landmark determinado se encuentra en dicha imagen. Cuando el landmark ha sido identificado, puede realizar su seguimiento mediante movimientos discretos del robot, manteniendo el landmark dentro del campo visual de la cámara. También puede informar sobre la distancia existente entre el landmark y la cámara a fin de determinar si está dentro de un radio que indica si hemos o no alcanzado un sitio del entorno en particular. En resumen, este comportamiento realiza tres funciones: detección del landmark, seguimiento del landmark y medición de la distancia entre el landmark y la cámara, las cuales se comentan a continuación:

Detección del landmark.

La detección del landmark se realiza mediante el conjunto de librerías ARToolkit. Empleando este sistema es posible detectar landmarks visuales de forma cuadrada que se componen de un cuadrado negro, en cuyo centro se tiene otro cuadrado color blanco cuatro veces más pequeño, y, en el interior de éste último, se dibuja un patrón, como lo ilustra la figura 4.1. La aplicación, utilizando las funciones proporcionadas por ARToolkit, será capaz de detectar este tipo de plantillas por medio de las imágenes de video capturadas.

Figura 4.1: Ejemplo de marca para ARToolkit

Una vez detectada una marca en una imagen, ARToolkit permite calcular la posición y orientación relativa de la marca respecto a la cámara. El funcionamiento es el siguiente:

- Primero se captura un fotograma del mundo real mediante la cámara.
- Se umbraliza la imagen de forma que los píxeles cuya intensidad supere el valor de un cierto umbral (threshold), son transformados en píxeles en color negro, mientras el resto se transforman en píxeles en color blanco.
- Se buscan todos los marcos negros como los del patrón capturado en la imagen de video. La acción de umbralizar la imagen hará que el marco negro aparezca como blanco, y el cuadrado blanco aparezca como negro.
- Se hace una comparación del patrón contenido en el interior del cuadrado blanco, con las plantillas de las que se tiene información almacenada.
- Si la forma del patrón analizado y la plantilla almacenada coincide, se utiliza la información de tamaño y orientación de la plantilla almacenada para compararla con la que se ha detectado y así poder calcular la posición y orientación relativas de la cámara al patrón, y se guarda en una matriz.
- La matriz se utiliza para establecer la posición y orientación de la cámara virtual (transformación de la vista), lo que equivale a una transformación de las coordenadas del objeto a dibujar.
- Al haber puesto la cámara virtual en la misma posición y orientación que la cámara real, el objeto virtual se dibuja sobre el patrón, se renderiza, y se muestra la imagen resultante, que consiste de la imagen del mundo real y el objeto virtual superpuesto, alineado sobre el patrón.
- El mismo proceso es llevado a cabo para las siguientes imágenes captadas.

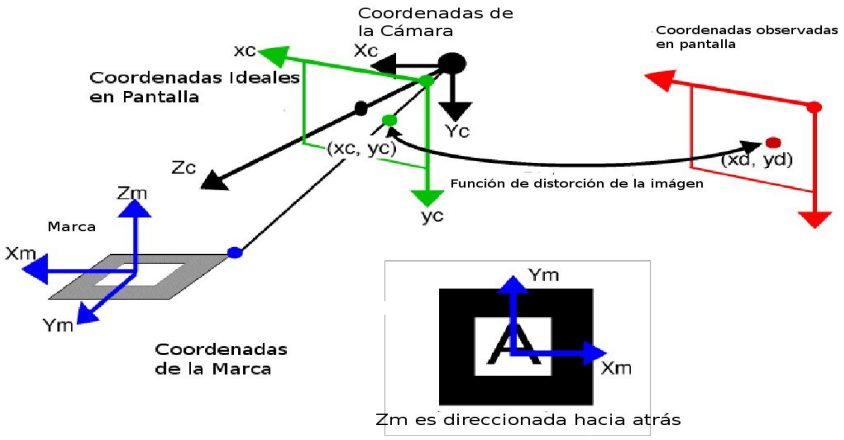


Figura 4.2: Sistema ARToolKit

Seguimiento del landmark.

Una vez que el landmark ha sido identificado se procede a su seguimiento, es decir, el objetivo es tener el landmark siempre dentro de la siguiente imagen a pesar del movimiento del robot. Para ello, el comportamiento *Navegar hacia el landmark*, posiciona al robot en cada desplazamiento para que la marca quede en el centro de la imagen. La información que se necesita para centrar el landmark es el ángulo que existe entre el centro del landmark y el centro de la imagen, y a partir de ahí, habrá que establecer la correspondencia necesaria con el movimiento de la base del robot (o del pan-tilt según sea el caso) para hacer girar al robot y quede alineado a un ángulo próximo a cero grados respecto al landmark.

Medición de la distancia cámara - landmark.

El navegador reactivo necesita conocer la distancia entre el robot y el landmark para determinar si esta lo suficientemente cerca como para considerarla alcanzada. Esta medición es posible realizarla directamente de la información que proviene del sistema ARToolkit sobre el landmark.

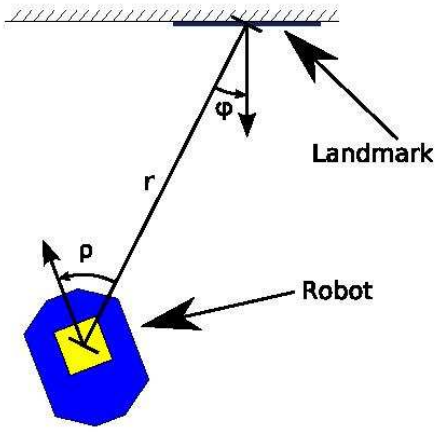


Figura 4.3: Ángulo del landmark respecto a la posición del robot.

4.4. Máquina de estados para Navegar mediante landmarks artificiales

La técnica de máquinas de estados es una útil herramienta para manejar secuencias de comportamientos. Generalmente las máquinas de estados se escriben en forma de *Diagrama de estados* que permite diseñar de forma gráfica lo que debe realizar el programa en un tiempo y condiciones determinadas.

Una máquina de estados está compuesta por un conjunto de comportamientos o estados. En cada estado se evalúa una serie de condiciones de entrada, para determinar si cumple una o más condiciones que le permitan hacer una transición de entre un conjunto de transiciones posibles.

A continuación se describe la máquina de estados para la navegación mediante landmarks artificiales presentada en la figura 4.4.

Estado 1: Buscar Landmark *Acción*: Este comportamiento consiste en hacer girar el pan-tilt o la base del robot según se requiera, en busca de un landmark específico.
Entradas: Solo una, el número del identificador del landmark, que proviene del planeador de rutas.
Transiciones: Si ocurre “Landmark no encontrado”, se hace la transición al comportamiento *Explorar por landmarks*. Si ocurre ”Landmark encontrada”, se hace la transición al comportamiento *Navegar hacia el landmark una distancia d*.

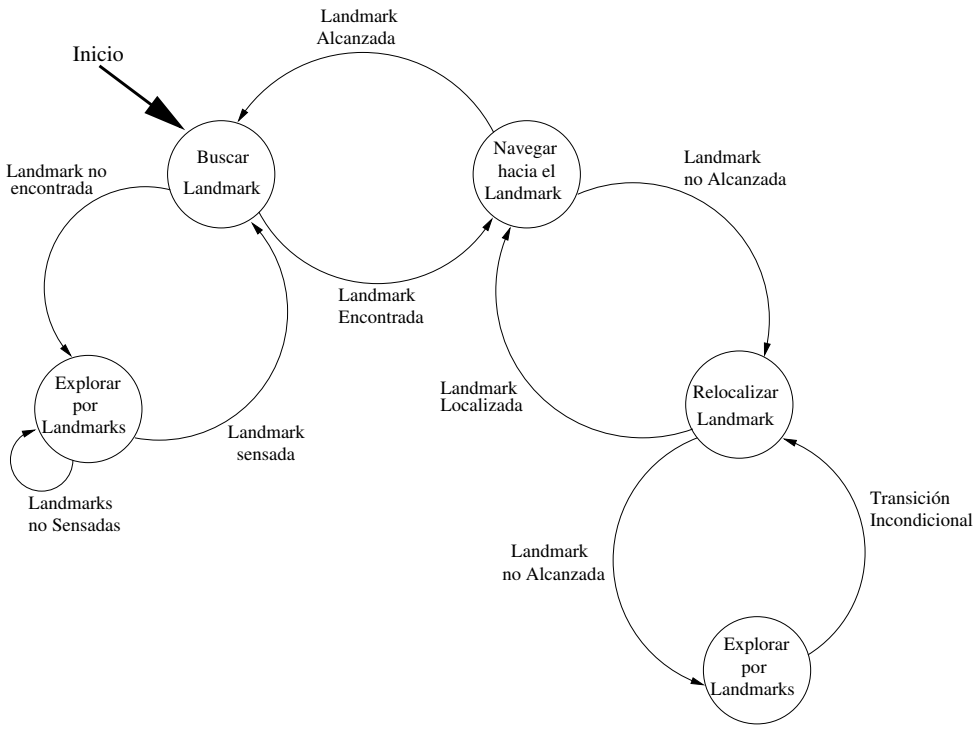


Figura 4.4: FSA para navegar mediante landmarks colocadas en el entorno.

Estado 2: Merodear por landmarks *Acción*: Este comportamiento consiste en deambular por el entorno, hasta detectar una o varias landmarks.
Entradas: Ninguna.
Transiciones: Si ocurre "Landmarks no sensadas", se permanece ejecutando el comportamiento "Merodear por landmarks". Si ocurre "Landmarks sensadas", va al comportamiento previamente visitado, ya sea *Buscar Landmark*, o *Relocalizar landmark*.

Estado 3: Navegar hacia el landmark *Acción*: En este comportamiento, el robot gira hasta orientarse al centro del landmark, y entonces avanza una distancia fija en dirección del landmark.
Entradas: Número de identificación del landmark
Transiciones: Si ocurre "Landmark no alcanzada", se hace la transición hacia el comportamiento *Relocalizar landmark*. Si por el contrario, ocurre "Landmark alcanzada", hace la transición al *estado de inicio*.

Estado 4: Relocalizar landmark *Acción*: Para este estado, el robot gira 45 grados en pasos de 15 grados, primero hacia un lado y después hacia el otro, cubriendo un área total de 90 grados, en busca del landmark requerido.
Entradas: Número de identificación del landmark.
Transiciones: Si ocurre "Landmark no localizado", se hace la transición hacia el comportamiento *Merodear por landmark*. Si por el contrario, ocurre "Landmark localizada", regresa al estado *Navegar hacia el landmark*.

Capítulo 5

Desarrollo

En este capítulo se aborda el desarrollo de la plataforma de software que sustenta este proyecto.

Bibliotecas de procesamiento de imagen

Las operaciones sobre la imagen es llevado a cabo a través de la biblioteca ARToolkit, un código de fuente abierta desarrollado por Hirokazu Kato de la Universidad de Hiroshima. Esta sistema implementa una variedad de funciones para reconocimiento de marcas visuales y generación de realidad aumentada. La biblioteca no solo proporciona operaciones para el procesamiento de las imágenes, sino también funciones de más alto nivel para la calibración de cámaras, entrenamiento de nuevos patrones visuales y seguimiento de objetos, etc.

Bibliotecas auxiliares

El sistema desarrollado emplea componentes software adicionales relacionados con componentes secundarios, tales como las comunicaciones, o la planificación de procesos. Resultan críticos algunos servicios básicos del sistema operativo, como los *threads* y *sockets*, entre otros. En el diseño del sistema se incluyeron las tareas de navegación, percepción y control ejecutándose como *hilos de procesamiento independientes*, y su comunicación se manejo mediante sockets UDP. Los sockets además se aplicaron en resolver las tareas de comunicación remota con la computadora de a bordo, así como con otros dispositivos de procesamiento externo.

Software de desarrollo propio

Por último, tanto los algoritmos propuestos en esta tesis como el software global del sistema tuvieron que ser codificados expresamente. En total, los procedimientos para el control de movimiento del robot, máquinas de estado para búsqueda de landmarks, entrenamiento y adaptación de ARToolkit, planeador de caminos, e interfaz de comunicaciones mediante sockets, fueron programados en C++, utilizando todas las bibliotecas comentadas y ocupando un total de unas 3,000 líneas de código.

5.1. Organización

La figura 5.1 muestra el diagrama a bloques de la arquitectura de hardware y software desarrollada.

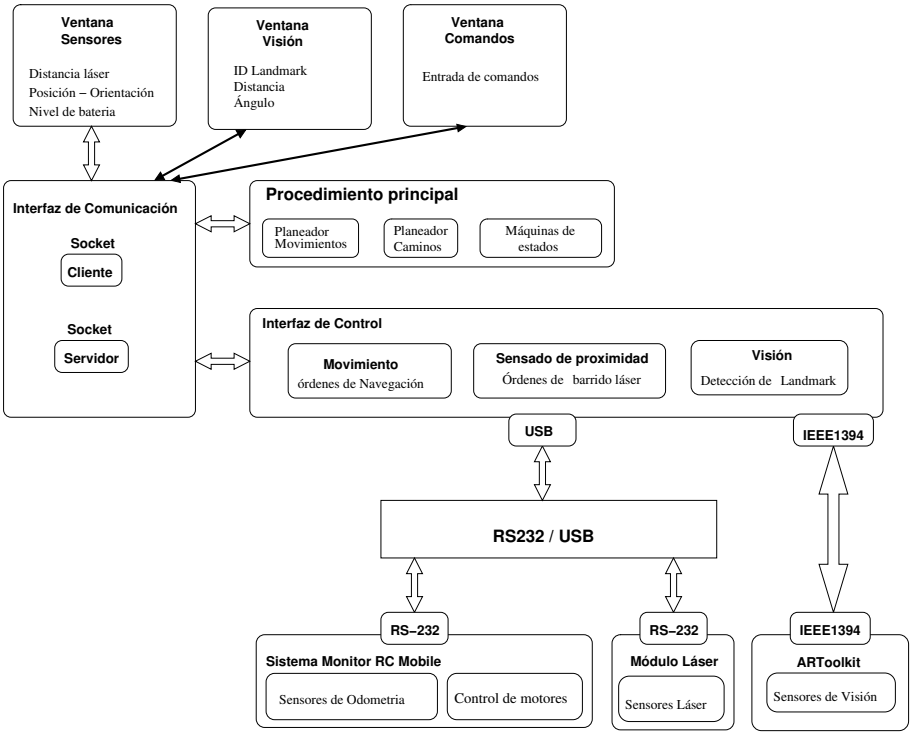


Figura 5.1: Arquitectura Software del sistema.

5.1.1. Arquitectura de procesos distribuida

Una vez conocidos en detalle los componentes utilizados, describiremos la organización del software desarrollado, así como su relación con el hardware del robot. El diseño de la arquitectura de procesamiento del robot, se separó en procesos independientes, para las tareas de lecturas de sensores, el control de los motores del robot, el procesamiento de la imagen y la interfaz de comandos de control. Estos se comunican entre sí a través de sockets. Esta arquitectura modular permite la *flexibilidad* y *extensibilidad* del sistema.

5.1.2. Rutinas de control de bajo nivel

Estas rutinas son las que interactúan directamente con el hardware del robot, tanto para controlar los motores que guían su movimiento, como para leer los sensores de odometría que

sirven para conocer el movimiento que efectúa el robot. Estas rutinas conforman el sistema monitor residente en la memoria del microcontrolador, y se mantiene en ejecución de modo independiente del resto del software.

Para mantener la compatibilidad en el formato de instrucciones utilizado en el sistema ViRbot [13], se desarrolló una interfaz que interpretase las instrucciones ViRbot a los comandos base del robot, dicha interfaz se ejecuta como un hilo de procesamiento independiente. La finalidad de procesar instrucciones en formato de ViRbot, es mantener la compatibilidad de instrucciones que utilizan el simulador de robots móviles *Roc2*, así como permitir probar los algoritmos y maquinas de estados en otras plataformas de hardware que ya admiten comandos ViRbot.

Entre las instrucciones que conforman esta interfaz, se habilitaron comandos de movimiento de las ruedas del robot, comandos de movimiento del pan-tilt, lectura del estado del sensor odométrico, estado de las baterías, lecturas de los sensores de láser y de visión. Como vía de comunicación con el microcontrolador, esta interfaz utiliza el puerto serie.

En el siguiente nivel de jerarquía, ejecutándose en la computadora de abordo, se encuentran los *procedimientos de alta prioridad*. Estas tareas están en contacto continuo con el microcontrolador del robot, a través de instrucciones transmitidas a la interfaz de comandos mediante sockets. En una de ellas, por ejemplo, el módulo de navegación reactiva solicita lecturas de los sensores de proximidad (en este caso del láser), para determinar la distancia hacia los obstáculos presentes en el entorno. Otra tarea prioritaria es el procesamiento de la visión para detección de landmarks artificiales, mediante el sistema ARToolkit. Finalmente, se incluye una tarea para llevar a cabo la odometría del robot (puesto que no se le contabiliza en el sistema monitor).

5.1.3. Procedimiento principal

En el nivel superior se encuentran la planificación del camino, las máquinas de estado, y la navegación reactiva. En este procedimiento, se controla la navegación del robot empleando landmarks visuales artificiales. Del cruce de datos provenientes de los sensores de visión, láser y odometría surge el ciclo de planificación de ruta, búsqueda de landmarks y navegación reactiva, propuestos en los capítulos 3 y 4.

A partir de la información del destino que se desee alcanzar (este dato lo aporta el usuario), se realiza un cruce de datos para determinar el nodo más cercano al robot a partir de su posición actual. Con esta información se efectúa el proceso de obtener el vector de nodos que conforman la ruta de menor coste desde la posición actual del robot hasta el destino, empleando Dijkstra. Posteriormente, empleando máquinas de estados se efectúa la detección de landmarks utilizando el sistema de visión como un sensor de landmarks, se busca el landmark asociado al identificador del nodo en cuestión, para determinar su posición. cada uno de los nodos devueltos por el planeador, y reactiva se guía al robot de forma discreta hacia cada uno de los nodos que conforman el vector, pero cuidando en todo momento no chocar con obstáculos inesperados.

Para ello, se comparan los landmarks detectados en el entorno, con los landmarks que conforman el mapa topológico.

Sobre la plataforma descrita en los capítulos anteriores, hemos implementado los algoritmos encargados de realizar los objetivos planteados en el la sección 1.2. Estos algoritmos y su implementación codificada en C++ se detallan en este capítulo.

5.2. Algoritmo de Dijkstra

Para implementar este algoritmo sobre el mapa topológico codificado en una gráfica, se asigna el coste de cada arista como la distancia euclídea entre sus dos vértices. Las coordenadas de cada vértice de la gráfica se tienen almacenados en una *tabla de conectividad* como la mostrada a continuación:

(connection	4	5	15.6)
(connection	5	4	15.6)
(connection	3	2	11)
(connection	2	3	11)
(connection	2	4	20)
(connection	4	2	20)
(connection	3	0	22.5)
(connection	0	3	22.5)
(connection	0	1	29)
(connection	1	0	29)
(connection	3	1	25)
(connection	1	3	25)

Figura 5.2: Tabla de conectividad.

Los datos de entrada de este algoritmo son:

- Vértice Origen
- Vértice Destino
- Tabla de conectividad
- Tabla de asociación de vértices con landmarks

Como vértices origen y destino tomamos la posición actual del robot y el destino especificado por el usuario.

(centroid 0	24.500000	7.000000)
(centroid 1	12.500000	44.500000)
(centroid 2	0.500000	28.000000)
(centroid 3	0.500000	15.600000)
(centroid 4	12.000000	0.500000)
(centroid 5	24.500000	7.000000)

Figura 5.3: Tabla de localización de los nodos.

(node	1	IMAGE1)
(node	2	IMAGE2)
(node	3	IMAGE3)
(node	4	IMAGE4)
(node	5	IMAGE5)

Figura 5.4: Tabla de asociación de vértices con landmarks.

8

La salida de este algoritmo es un vector de nodos que se corresponden con el camino mínimo encontrado entre los vértices de origen y de destino, siempre y cuando la gráfica sobre la cual se aplique el algoritmo sea conexa.

5.3. Comunicación mediante sockets

Los sockets empleados fueron los *UDP* y su aplicación en el proyecto se derivo de la librería msocket.h, que fue incluida para establecer la comunicación entre todos los procesos que intervienen en la arquitectura de software propuesta.

El tamaño de los buffers de transmisión y recepción fue definido a un valor máximo de 1024 bytes.

5.4. ARToolkit

Se modificó el código de ejemplo *simpleTest.c* incluido en su compilación base para contar con las funciones que permiten estimar la distancia y ángulo de la marca detectada. También se incorporó en el loop principal de este código, la utilización de sockets para el envío de los datos *ID, Distancia, Ángulo* de los landmarks detectados. Para la recepción de comandos de control se definieron instrucciones que permiten enviar mediante sockets, información acerca

de los landmarks, y, también se agregaron comandos para solicitar el sensado por visión del medio ambiente.

Se seleccionaron 17 patrones creados para el sistema ARTag para ser reconocidos por ARToolkit, utilizando la utileria *mkpatt* que forma parte del sistema ARToolkit y cuyo funcionamiento se detalla en el apéndice A.

5.5. Navegación reactiva

Asumiendo que el movimiento del robot consta de un ciclo de sensar y avanzar, si éste no utilizase un sistema de evasión de obstáculos, el robot sencillamente se perfilaría hacia su objetivo siguiendo una trayectoria en línea recta.

Sin embargo, contando con el método como el presentado en el capitulo 3, el vehículo, en lugar de avanzar directamente hacia una meta, el ciclo de sensar-avanzar se plantearía como el avance por sub-metas cumpliendo hasta donde sea posible con la trayectoria global, y siguiendo una trayectoria no necesariamente en línea recta hacia la meta.

En la implementación del algoritmo de campos potenciales debe contarse con los siguientes datos de entrada:

- Posición actual del robot.
- Posición de los obstáculos.
- Posición del objetivo.
- Valor del ajuste del radio de seguridad.

La salida de este algoritmo es el siguiente *punto coordenado de avance*.

El desarrollo consta de los siguientes pasos:

- **getScan(posIni, posFin)**: recibe un rango de apertura del haz del láser, comprendido entre la posición inicial y final del haz, y devuelve las lecturas en dichas posiciones.
- **get-repulsion-force(arraySensor, angleSensor, angleRobot)**: recibe las lecturas de un conjunto de sensores (láser, sonar, visión), con un ángulo respecto al centro del robot, así como el ángulo del robot respecto al plano coordenado, y devuelve el vector de fuerza total de repulsión de acuerdo a la fórmula (3.1).

- **get-attraction-force(coordRobot, thetaRobot, coordDestiny)**: recibe la posición y orientación del robot en el plano coordenado, y las coordenadas del punto o destino a alcanzar, y devuelve el vector de fuerza total de atracción hacia dicho punto.
- **addvectors(repulsionVect, atractionVect)**: recibe los vectores de atracción y de repulsión, y devuelve su adición, que representa el vector de movimiento que corresponde a la fuerza total que ejercen tanto las fuerzas atractivas de las submetas, como las fuerzas de repulsión de los obstáculos.
- **get-advance-angle(mv-vector)**: recibe un vector *distancia, ángulo*, y devuelve las coordenadas del siguiente punto de avance.

Capítulo 6

Resultados

En los capítulos anteriores, hemos descrito los métodos empleados en esta tesis para la navegación de un robot empleando landmarks artificiales. En este capítulo examinaremos el resultado de la aplicación de dichos métodos a través de pruebas sobre el sistema de navegación aplicados en el robot RC-Mobile. Primero describimos la implementación y los supuestos relacionados a la preparación de nuestros experimentos. Después, presentamos los experimentos que hemos desarrollado para evaluar las pruebas siguientes:

1. Posicionamiento empleando odometría: Que tan fiable es el posicionamiento del robot utilizando solamente sus sensores de odometría para la navegación.
2. Navegación reactiva: Evalúa la habilidad del robot para alcanzar metas dentro del mapa topológico, así como evitar chocar con obstáculos presentes en el entorno.
3. Navegación con landmarks: Se prueba la capacidad del robot para navegar empleando como sensor de los objetivos a alcanzar, un sistema de visión de detección de marcas visuales.

6.1. Supuestos

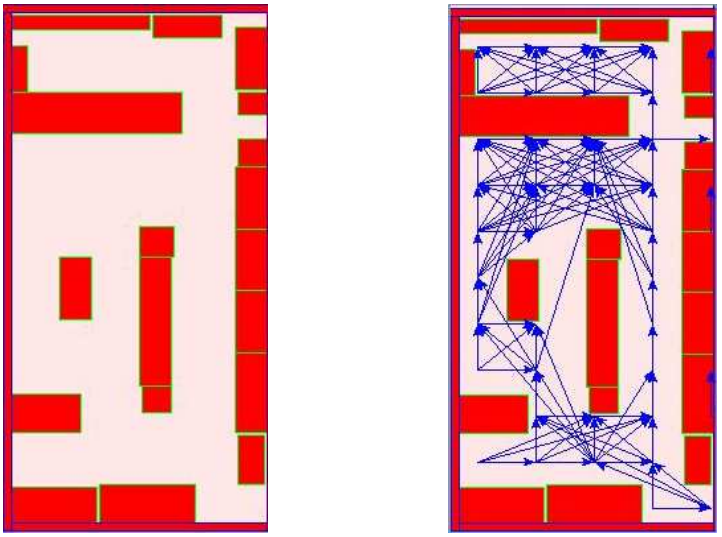
Asumiremos que se cuenta de antemano con el mapa topológico del entorno, y el espacio de navegación se considera planar, y finalmente que los obstáculos se representan mediante polígonos. En el mapa se consideran aquellos obstáculos medianos y grandes, tales como escritorios, sillas, y obstáculos fijos como paredes y pasillos. No se considera la altura del robot y cualquier obstáculo que pudiese colisionar con la parte superior del robot.

6.1.1. Entorno de navegación

El entorno de navegación es representado por polígonos: uno representa los límites del entorno, y otros son utilizados para representar obstáculos dentro del entorno. El mapa del entorno (ver figura 6.1) se construyó midiendo el interior del laboratorio. Éste tiene un área

total de 35 metros cuadrados y en su interior tiene instalados varios escritorios, sillas y objetos de tamaño considerable, de los cuales se conoce su posición, y se asumen fijos.

Para generar el mapa geométrico hemos utilizado un cuantizador vectorial [14] desarrollado en el laboratorio de Bio-Robótica, mediante el cual se dividió el entorno en pequeñas regiones, de cuyos centroides se obtiene el par coordenado x,y en el espacio euclideano, al cual corresponde un nodo de de la gráfica que representa al mapa topológico.



(a) Mapa métrico del entorno

(b) Mapa topológico del entorno

Figura 6.1: El entorno interior de pruebas fue el laboratorio de Bio-Robótica de la FI-UNAM. (a) Presenta el mapa métrico construido manualmente, y en (b) el mapa topológico generado a partir del mapa métrico.

6.2. Pruebas y ajustes

6.2.1. Medición de errores odométricos

Esta sección describe los resultados de la medición de errores odométricos poniendo énfasis en la medición de los errores angular y de desplazamiento. Lo que se desea es medir el nivel de afectación que aporta el considerar únicamente la odometría para el seguimiento de caminos.

Error de traslación

Para determinar la fiabilidad de los datos odométricos, se realizaron pruebas para desplazar al robot de un punto de partida hacia una meta sobre el eje coordenado x. Dichas metas fueron establecidas a distancias desde los 0.3 metros hasta los 4 metros. Después de cada prueba se midió la posición real del robot mediante un sensor de láser. El error corresponde a la diferencia que resulta del valor real medido mediante el láser respecto a la medición hecha por la interfaz odométrica.

Se programo una aplicación para que el robot hiciera un número de pruebas indicado por el usuario de forma autónoma y en un tiempo reducido. En total se realizaron 423 pruebas. La prueba se realizó del siguiente modo:

Se envían comandos al robot secuencialmente con instrucciones de desplazamiento traslacional, y por cada vez que el robot informó haber alcanzado la posición indicada, se midió su desplazamiento real empleando el sensor láser y se registró el valor leído junto a la medición odométrica en un archivo de datos. Para evitar acumular errores de deslizamiento de las ruedas o de avance no uniforme, se colocaron landmarks visuales en los extremos frontal y posterior perpendiculares a la trayectoria que debe seguir el robot. Con estos landmarks y después de haber alcanzado cada meta programada, se aplica el comportamiento *buscar landmarks* descrito en la subsección 4.4 para detectar el landmark adecuado según la orientación del robot, y una vez detectado se obtiene el ángulo de posición respecto al robot. Este dato sirve para dar la orden al robot de rotar en la misma magnitud, buscando posicionarse a cero grados respecto al landmark, pero como la rotación también acarrea errores odométricos, se repite la acción de sensar el landmark y hacer girar al robot, hasta que el error sea menor a un valor muy pequeño, para nuestro caso seleccionamos el valor de $0{,}9 grados$ que al no ser muy grande permitía al robot en pocos pasos alcanzar a posicionarse a casi cero grados respecto al landmark. Una vez que el robot está alineado, se procede a repetir el ciclo *avanzar-medir-registrar-reorientar* hasta haberlo efectuado el número de veces indicadas por el usuario. Los resultados obtenidos se muestran en la figura 6.4-b.

Las gráficas obtenidas en esta prueba indican que el error de traslación es pequeño para desplazamientos cortos de entre -20 y $20dm$. En las pruebas se observó que la distancia real de desplazamiento era distinta a la indicada, debido a que el sistema de frenado era aplicado con una intensidad mayor en una de las ruedas y que dejaba al robot orientado a un ángulo distinto al ángulo de partida.

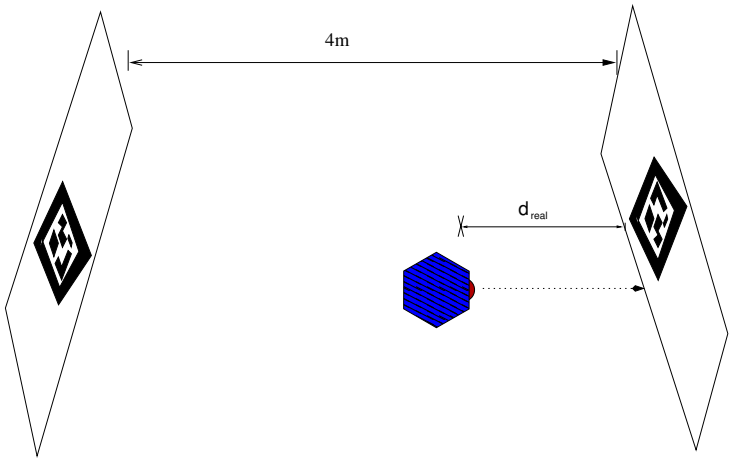


Figura 6.2: Prueba para medir el error de traslación: Mediante el sensor de láser se mide la distancia real respecto a la distancia esperada dada como orden. Se colocan dos landmarks al frente y detrás del robot, para que los desplazamientos se realicen perpendicularmente a la cara del landmark que este al frente en el momento de la prueba y corregir en cada prueba los errores de odometría debidos a rotación.

Error de rotación

Para esta prueba se habilitó una máquina de estados que hiciera girar al robot en distintos ángulos, y mediante un par de medidas del sensor láser obtuviera el ángulo real de giro de cada movimiento de rotación. Para evitar acumular errores odométricos después de cada orden de rotación dada, se siguió un procedimiento parecido al visto antes, en la cual se colocó un landmark a cero grados frente al robot sobre una superficie plana en la que además incide el haz del láser, y se hace girar al robot hasta posicionarse a un valor muy cercano a cero grados respecto al landmark (ver figura 6.3).

El procedimiento es el siguiente:

- Primero se hace girar al robot hasta posicionarse a cero grados respecto al landmark posicionado al frente del robot.
- Se mide con el láser la distancia a la superficie vertical uniforme puesta frente al robot, la cual se toma como el cateto adyacente.
- Se da la orden de girar al robot un ángulo arbitrario, pero cuidando que el láser proyectado no se salga del área de la superficie plana. Este valor corresponde a la hipotenusa.
- Finalmente se obtiene el ángulo de giro real aplicando la identidad trigonométrica:

$$theta_{real} = cos^{-1}(catetoadyacente/hipotenusa) \tag{6.1}$$

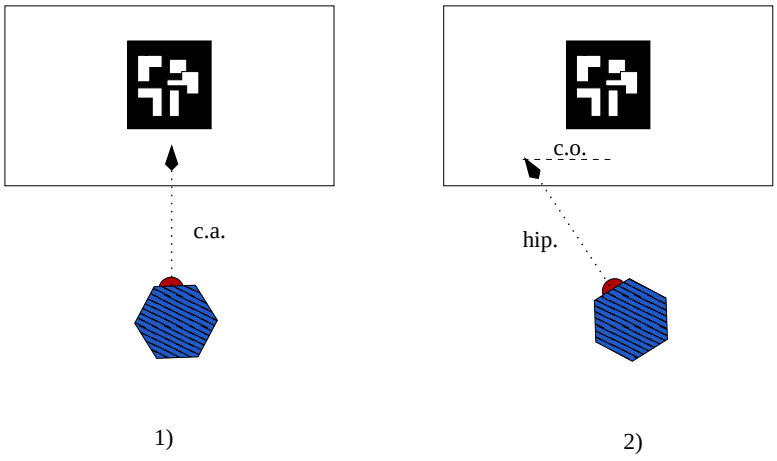
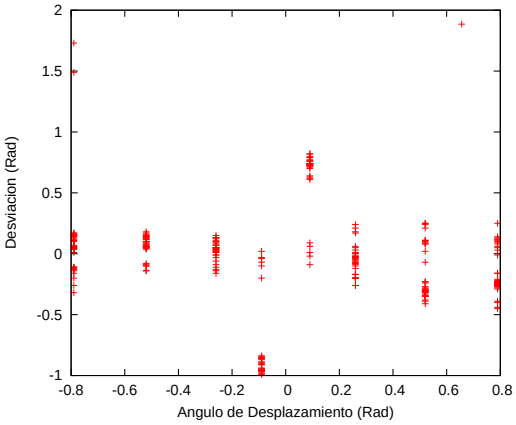
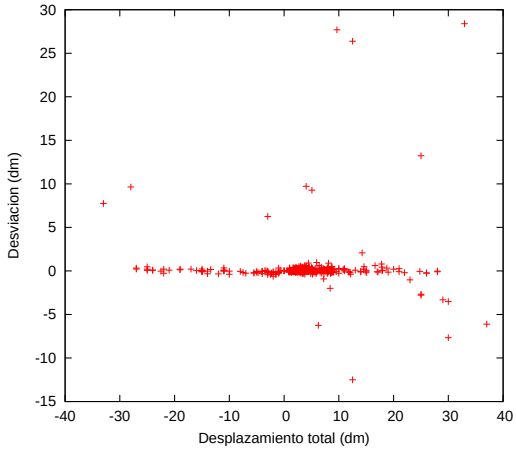


Figura 6.3: Esta prueba utilizó un sensor láser para medir el cateto adyacente (1) y la hipotenusa (2) y entonces realizar el cálculo del ángulo formado por ambas lineas obtenidas. Una marca geométrica se coloca para posicionar al robot casi perpendicularmente respecto al plano de la marca, y para ello se emplea el sistema de detección de marcas.

El error observado en estas pruebas muestra que para movimientos de rotación de entre $-0{,}2$ y $0{,}2rad$, el error es considerable: $-0{,}3$ hasta $0{,}9rad$, y este error acumulado en el recorrido de un robot representa un problema importante de posicionamiento.



(a) Resultado del experimento para medir el error rotacional odométrico. Los valores de giro positivos representan giros hacia la izquierda, y aquellos negativos representan giros hacia la derecha.



(b) Resultado del experimento para medir el error traslacional odométrico. Se observa que para valores menores a 20 dm. el error es pequeño, pero conforme rebasa este valor el error también crece sustancialmente

Figura 6.4: Calidad de la Odometría

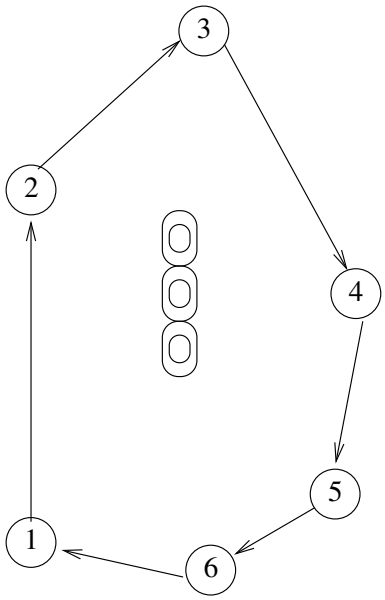
6.2.2. Navegación en el entorno sin landmarks

Esta prueba consistió en seguir una trayectoria semi circular por el entorno de tal forma que el robot volviera al punto de partida. El camino recorrido es el que se muestra en la figura 6.5. Según lo demuestran las 13 pruebas realizadas, en 5 de ellas el robot se acercó al destino con un error promedio de 4.7 decímetros, y el resto de las veces, el robot no llegó al destino por el acumulamiento de errores de frenado que al estar de manera normal ajustado para frenado suave, se pasaba del punto en el que debería de frenar y quedar estático, lo que acumulaba errores de odometría que hicieron al robot colisionar con los obstáculos presentes.

Ajuste de parámetros

Para reducirlos errores se ajustó el valor del parámetro de aceleración de los motores acoplados a las ruedas de la base. Uno de los parámetros que se observó más influyen en el frenado del robot fue el parámetro T_h que modifica el comportamiento cuando las ruedas alcanzan la posición deseada y se activa el frenado magnético. El frenado magnético ocasionado por la circulación continua de corriente por el motor, permanece por T_b milisegundos después de que se alcanza la posición deseada. T_b tiene que tener un valor pequeño para no consumir mucha energía de las baterías. Al vencer este tiempo se incluye la resistencia de frenado del circuito del motor por T_h milisegundos. Concluido dicho intervalo de tiempo, el motor se apaga y mientras no se contabilicen V_h unidades del encoder, el motor continúa apagado. Si el encoder alcanza las V_h unidades se vuelve a activar el frenado magnético por

(a)



(b)

Figura 6.5: Entorno de navegación utilizando solamente las medidas odométricas. Ejecución de una prueba en el laboratorio de bio robótica en (a) y un camino obtenido del mapa topológico (b).

T_h milisegundos y el proceso se repite indefinidamente hasta alcanzar el reposo total del robot [12].

Experimentalmente encontramos un valor de $T_h = 3000$ y de $T_b = 500$ que permitió un frenado suave pero firme lo se mejoró el posicionamiento en las sub metas reduciendo el error, y esto se vio reflejado al obtener 5 de las 13 pruebas en las que el robot llego a una distancia próxima a la meta sin colisionar con los obstáculos presente en el espacio de prueba.

6.2.3. Navegación reactiva

Este comportamiento se ocupó para realizar la navegación del robot por el entorno, relacionando la información sensorial (visión y láser) con las órdenes dadas a los actuadores del robot en función del desplazamiento de una posición hacia otra, y colocando en medio del camino dos obstáculos (ver figura 6.6).

La primer prueba fue verificar la habilidad del robot para evadir obstáculos. Se realizaron 17 pruebas en el escenario mostrado en la figura 6.6, y al ajustar el radio de seguridad así como la magnitud de los potenciales de repulsión y de atracción fue posible conseguir que el robot no colisionará con el obstáculo en 14 de las 17 pruebas.

Podemos afirmar entonces que el robot mostró un buen desempeño para la evasión de

obstáculos, no obstante que en ocasiones llego a colisionar con obstáculos, pero ajustando las constates de repulsión a $K = 0{,}5$, la constante de atracción $A_t = 240{,}05$ y el radio de seguridad a $9dm$ (subseccion 3.3.1) pudo mejorarse el resultado.

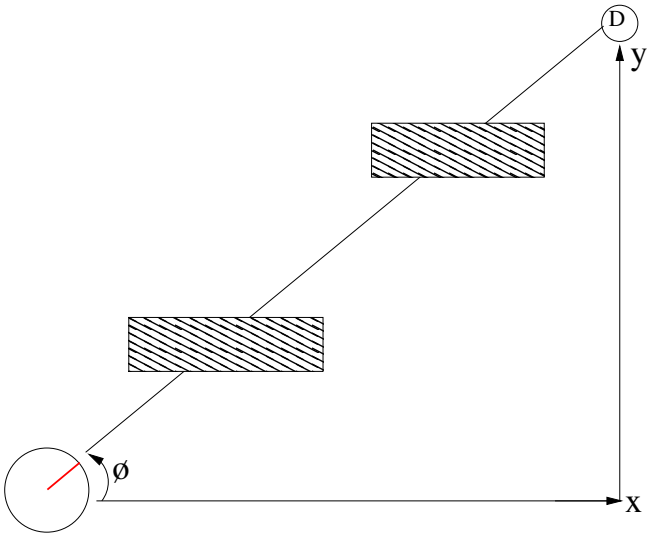


Figura 6.6: Prueba para la evasión de obstáculos: dos obstáculos colocados en diagonal se emplearon para probar la evasión de obstáculos mediante el método de campos potenciales.

6.2.4. Navegación utilizando landmarks

Para realizar esta prueba se colocaron en el entorno 6 landmarks artificiales como las que se muestran en la figura 6.7. Según la posición de cada landmark, se fijo su posición dentro de la tabla de conectividad. Una vez hecho esto, se inicia el ciclo siguiente:

- Obtener del planeador la ruta a seguir de acuerdo al mapa topológico.

- A continuación, se ejecuta el plan.

- De acuerdo al camino obtenido en el primer punto, el robot realiza la búsqueda de la marca que corresponde con el identificador del nodo obtenido del resultado del planeador, utilizando la cámara de a bordo montada sobre el pan tilt.

- Suponiendo que el robot ha identificado a la landmark que corresponde a la solución del planeador, el robot debe aproximarse en pasos hacia ella, orientando la base en la dirección del ángulo obtenido mediante el sistema de visión.

  Una vez que el robot se encuentre a una distancia próxima a la ubicación del landmark, este proceso termina.

- Una vez alcanzado el landmark en cuestión, se solicita al planeador proporcione la siguiente meta a alcanzar.

(a)

(b)

Figura 6.7: Entorno de navegación utilizando landmarks visuales. (a) vista desde la parte trasera y (b) vista desde la parte delantera.

Capítulo 7

Conclusiones

El objetivo de este trabajo fue habilitar la navegación de un robot móvil dentro de un entorno interior, utilizando landmarks artificiales como guía principal en el diseño. Suponemos que este objetivo ha quedado cubierto en gran medida gracias a la implementación de los distintos componentes que conforman nuestro desarrollo, de las cuales mostramos a continuación las propuestas fundamentales:

Se implementó un planeador global mediante el algoritmo de Dijkstra para encontrar los caminos mínimos dado un mapa topológico del entorno.

Se desarrolló un conjunto de máquinas de estados para la búsqueda y detección de landmarks, utilizando para ello el conjunto de librerías de ARToolkit. Este grupo de librerías nos permitió modificar el código para agregar un sin número de patrones a detectar. La información extraída de la detección del landmark resulta de gran utilidad para la navegación precisa por el entorno captado en un mapa topológico.

Se ha desarrollado un comportamiento de navegación reactiva, que facilita al robot navegar por entornos desconocidos, pero sobre todo, evita que éste colisione con obstáculos no previstos por el planeador de caminos.

Finalmente, los componentes de software y de hardware empleados en el desarrollo se integraron en una arquitectura que permite ser extensible a través de incorporar nuevos comportamientos construidos a base de máquinas de estados.

En resumen, se han alcanzado los objetivos planteados en la introducción (sección: 1.2).

7.1. Desarrollos futuros

Una mejora al método de navegación consistiría en considerar dentro de la planeación de caminos la planificación de velocidades que permita a un robot desplazarse por el entorno a la mayor velocidad considerando las propiedades cinemáticas del robot y del camino.

Respecto a la ejecución de la planeación se propone utilizar un método de navegación mediante *splines* que suavizaría la trayectoria del robot.

En los aspectos relacionados con la implementación existen varias posibilidades como la implementación en hardware de las etapas de más bajo nivel: Navegación reactiva, ejecución de de máquinas de estados, detección de landmarks, hasta tareas más complejas como la construcción del mapa del entorno, empleando por ejemplo, un FPGA.

Capítulo 8

Apéndice A. Pataforma de desarrollo

Antes de explicar la implementación del proyecto, se detallan en este capítulo las herramientas de hardware y de software sobre las que nos hemos apoyado para su consecución.

Componentes de Hardware

El robot RC-Mobile Base-1

El robot disponible es el RC-Mobile Base-1 construido para navegación en entornos de interiores. Este robot lo comercializa la empresa Cervantes Co. [12]. El robot (figura 8.1) dispone de un sistema de giro inclinación (pan-tilt) que permite dar un giro completo y alojar sobre él, un dispositivo láser o una cámara de visión. Los actuadores del robot son cuatro motores de corriente continua, dos asociados cada uno a una rueda, y dos asociados al pan-tilt. Además dispone de dos ruedas multidireccionales (locas) colocadas detrás y al frente de la base, que unidas al movimiento que realizan los dos motores, favorecen notablemente los giros del robot. También lleva incorporados unos odómetros (*encoders*) asociados a las ruedas para la medición de los desplazamientos y giros efectuados por el robot.

En cuanto a los sensores de proximidad se hizo necesario incorporar sobre la base del robot un sensor láser, y sobre el pan tilt se montó una cámara de visión para la detección de marcas visuales.

Finalmente se incorpora una computadora portátil al robot, para ejecutar los programas de control, de navegación y de visión. Esta computadora esta conectada a una red local mediante un enlace inalámbrico, utilizando una tarjeta de red 802.11 que le proporciona una velocidad de hasta 100Mbps en sus comunicaciones. De esta forma el programa de control puede correr a bordo de la computadora portátil siendo consultado desde un tercer computador conectado a la mismo enlace de red.

El robot móvil tiene 45 cm de diámetro en la base, 112 cm de altura, y un peso de 53 kg... En cuanto a las características de carga eléctrica, el robot dispone de 4 baterías recargables

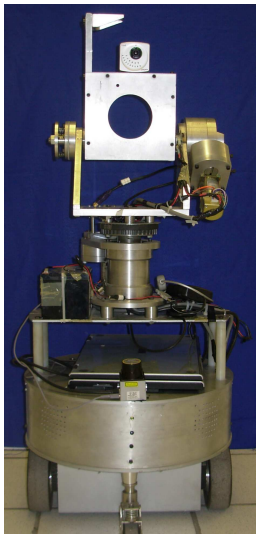


Figura 8.1: Robot de prueba RC-Mobile Base-1, éste robot posee un sensor láser y una cámara de visión para sensar el entorno.

de 12V que le dan una autonomía de funcionamiento de aproximadamente 1 hora.

El robot alberga dos microcontroladores 9S12DP256C de la familia HC12 de Motorola, con 256Kb de memora flash EEPROM cada uno, en la que tiene codificadas las rutinas de control de bajo nivel para la operación de los actuadores del robot, y manejo del sensor de odometría. La comunicación local entre el microcontrolador y una computadora externa, se establece mediante una interfaz serial RS-232. Una descripción más detallada de las características del RC-Mobile-Base-1 y de sus distintos componentes puede encontrarse en el manual de descripción del hardware del robot, que se proporciona en la bibliografía.

Sensores de Imagen

Las cámaras de visión empleadas son los modelos Logitech QuickCam Zoom y la Unibrain Fire-I. Cada uno de estos sensores ópticos van montados sobre el pan-tilt del robot, pero nunca se usan simultáneamente, ya que solo se emplean una a la vez para para hacer diferentes pruebas.

Cámara Unibrain Fire-i IEEE 1396

Es una cámara de sensor CCD (Charge Coupled Device) apta para uso en hogar y oficina. La buena calidad de imagen que produce y su bajo consumo de energía son factores convenientes para utilizarla en la detección de marcas visuales en entornos interiores, como lo muestran su aplicación en sistemas descritos en [18]. La cámara puede alimentarse de energía eléctrica desde su conector FireWire de 6 pines, o utilizando un conector de entrada que suministre una carga de 12 VDC. Para llevar a cabo la interfaz con la computadora

Figura 8.2: Cámara Unibrain Fire-i.

Figura 8.3: Cámara Logitech QuickCam Zoom.

portátil, se utilizo un conector Firewire y por separado se suministro la carga eléctrica desde una bateria de 12VDC recargable. Esta cámara obtiene imágenes claras y nítidas, y a tasas muy rápidas, permitiendo capturar resoluciones de vídeo VGA (640x800 píxeles) a una tasa de 30 cuadros por segundo. El flujo de vídeo provisto por una computadora portátil puede alcanzar una tasa de hasta 400Mbps a través de un puerto IEEE 1394a.

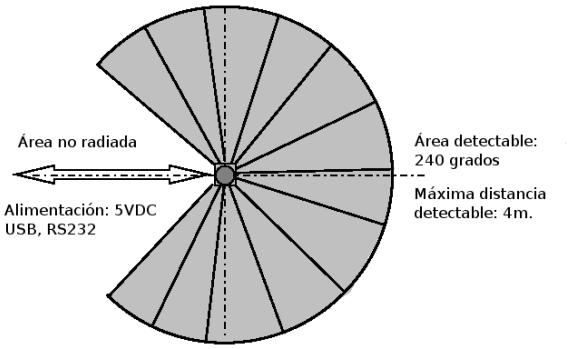
Cámara Logitech QuickCam Zoom USB2

La cámara Logitech QuickCam Zoom (figura 8.3) es una cámara pequeña y ligera, utilizada como cámara alternativa en la realización de pruebas. Esta cámara captura vídeo a una resolución máxima de 640 x 480 píxeles, a 40 cuadros por segundo (fps). La interfaz que utiliza es la USB, y el suministro de carga eléctrica lo obtiene desde el puerto USB de la computadora portátil.

Sensor Láser Hokuyo URG-04LX

El sensor láser URG-04LX de la marca Hokuyo, se comercializa para uso industrial y en aplicaciones de oficina. Su rango de detección angular de 240 grados, la distancia de de alcance de 4 metros, su bajo consumo de energía, y la resolución de 0.35 grados, lo hacen apropiado para detección de obstáculos en aplicaciones de navegación de robots móviles dentro de entornos interiores. Este sensor se muestra en la figura 8.4. El sensor debe alimentarse con +5 volts de corriente continua, y utiliza una interfaz serial RS232, o USB para comunicarse con un computador.

(a) Aspecto del láser.



(b) Área detectable del sensor: puede sensar objetos de hasta 70mm x 70mm. La distancia de detección varia según el tipo de objeto a sensar.

Figura 8.4: Sensor Láser de Rango Hokuyo URG-04LX

Hardware Comunicaciones

La tarjeta de red de la PC portátil montada al RC-Mobile-1, permite enlazar vía inalámbrica con la red local que permite un ancho de banda máximo de 100 Mbps. La conexión se realiza a través de un router Linksys que se conecta a cualquier punto de acceso de la red.

Componentes de Software

Sistemas Operativos

El controlador del robot está gobernado por un sistema monitor de tamaño reducido (menos de 32KB), que esta contenido en la EEPROM programable situada en el microcontrolador interno. Este pequeño sistema monitor ofrece al programador un conjunto de instrucciones para dar órdenes de movimiento de la base y del pan-tilt, así como órdenes para leer los sensores de odometría. El sistema monitor permite modificar ciertos parámetros de control de velocidad de los motores. El resto del software, ejecutado tanto en la PC portátil de a bordo, como en computadoras externas, funciona sobre distintas versiones del sistema operativo Linux, en distribuciones Ubuntu para arquitecturas de Intel.

Lectura de sensores y control del robot

La comunicación entre el microcontrolador y el PC se realiza mediante el puerto serie, a través del cual el programador envía órdenes y recibe la respuesta del estado del sistema monitor, usando un protocolo diseñado por los fabricantes. Este software se ejecuta sobre Linux, puede realizar la lectura de los sensores y dar las órdenes a los actuadores a través de una biblioteca de control para envío de mensajes construida sobre clases de C++ como parte de la labor de esta tesis, con la finalidad de liberar al programador de la necesidad de tener que conocer las interioridades de dicho protocolo.

Bibliotecas para el procesamiento de imagen: ARToolkit

ARToolkit es un conjunto de librerías de C/C++ que sirven para la creación de aplicaciones de realidad aumentada. Proporciona una serie de funciones para la captura de video así como para hacer una búsqueda de ciertos patrones en las imágenes capturadas, mediante técnicas de visión por computadora. También proporciona una serie de utilidades de gran ayuda al programador que quisiera realizar este tipo de aplicaciones.

Se abordará en detalle el funcionamiento de estas librerías en la implementación para la navegación mediante landmarks tratado en el capítulo 4.

Sockets

De modo muy simple se puede decir que un socket es una manera muy general de establecer comunicación con otras computadoras usando descriptores de archivos estándar de Unix. Los datos asociados con el socket son la dirección IP, y los números de puerto para ambos lados de la conexión, así como el protocolo de transporte (i.e. UDP o TCP) de la conexión. En Unix, todas las acciones de entrada y salida son desempeñadas escribiendo o leyendo en uno de estos descriptores de archivo, los cuales tienen como identificador un número entero, asociado a un archivo abierto que puede ser una conexión de red, una terminal, o cualquier otro proceso. Los sockets son bidireccionales, lo cual quiere decir que en ambos lados de la

conexión se pueden enviar y recibir datos. Sobre los distintos tipos de sockets de internet existen varios tipos de socket. Los sockets utilizados en esta tesis son los *sockets de flujo* (sock stream). Se se le llama *cliente* a la aplicación que inicia la comunicación y *servidor* la otra.

Sockets de flujo

Se caracterizan por considerarse libres de errores, por ejemplo, si enviáramos por el socket de flujo tres objetos **A, B, C** llegarán al destino en el mismo orden **A, B, C**. Estos sockets usan el protocolo TCP (Transmission Control Protocol) que nos asegura mantener el orden de los objetos durante la transmisión.

La aplicación de sockets en esta tesis se utilizó para comunicarse localmente entre los procesos, y externamente para controlar y monitorear el estado del robot y de sus tareas.

Clientes y Servidores

La tarea de los servidores es ofrecer acceso a la interfaz de control, y a los sensores de láser y de visión. La arquitectura de los servidores nace de la limitación de que al ejecutar cualquier programa en el robot que utilizase los sensores o los motores, fuera necesario hacerlo directamente en la computadora portátil conectada al robot. Esta arquitectura software se muestra en la figura 8.5.

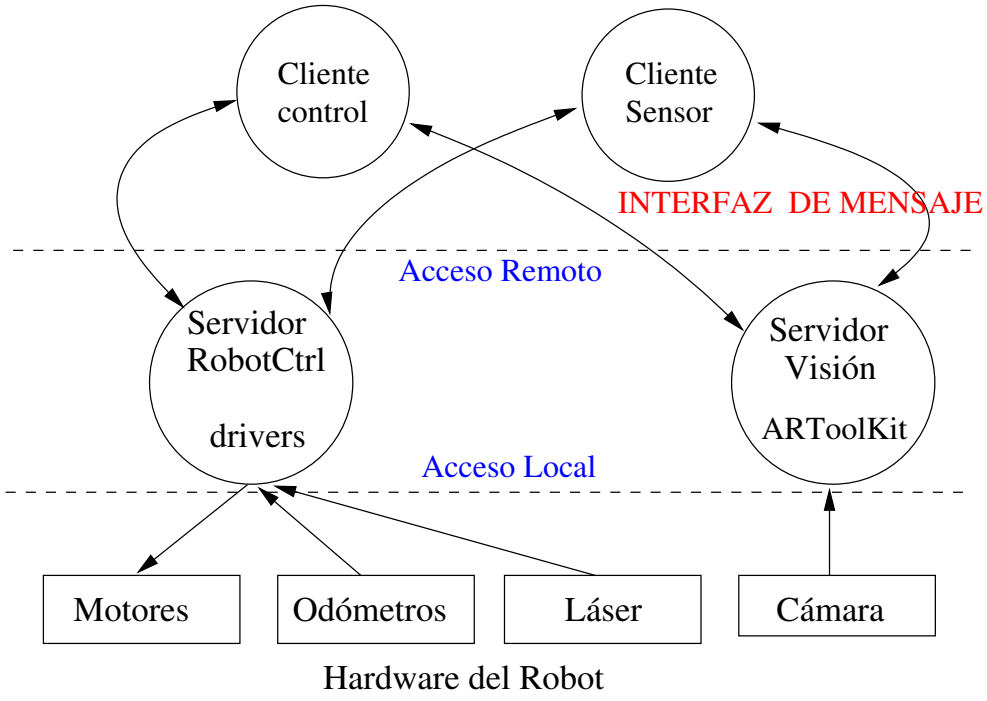


Figura 8.5: Arquitectura de clientes y servidores para el control del robot.

Bibliografía

[1] Ronald C. Arkin. *Behavior Based Robotics*. MIT Press, 1998.

[2] Kuipers B and Byun Y. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomoues Systems*, 8:47–63, 1991.

[3] J. Borenstein and Liqiang Feng. Correction of systematic odometry errors in mobile robots. *International Conference on Intelligent Robots and Systems*, 3, 1995.

[4] Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, and Sebastian. *Principles of Robot Motion*, chapter 1. MIT Press, 1998.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 1990.

[6] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. Technical report, University of Oxford, UK., 2003.

[7] A. Elfes. Sonar-based real-world mapping and navigation. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988]*, Junio 1987.

[8] L. Feng, J. Borenstein, and B. Everett. Where am i? sensors and methods for autonomous mobile robot localization. Technical report, The University of Michigan, 1994. disponible via anonymous FTP desde ftp.eecs.umich.edu/f/people/johannb.

[9] Kato H. and Billinghurst M. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. *2nd Int'l Workshop on Augmented Reality*, pages 85–94, 1999.

[10] Jean Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[11] Mataric M. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8:304–372, 1992.

[12] Leonardo Romero. Robot movil rc-mobile-base-1 + rc-pan-tilt-1. Technical report, Universidad de Michoacan, MÃ©xico, 2005. Desarrollado por Cervantes Co. Atzimba 168, centro, C.P. 60250, Paracho MichoacÃ¡n, MÃ©xico Tel.(423) 52511039.

[13] Jesus Savage, Adalberto Llarena, Gerardo Carrera, Sergio Cuellar, David Esparza, and Yukihiro Minami. Virbot: A system for the operation of mobile robots. *Robocup@Home*, 2006. http://www.ai.rug.nl/robocupathome/.

[14] Jesus Savage, Marco A. Morales, Adalberto Llarena, Sergio Cuellar, and Fernando Lepe Casillas. Construction of roadmaps using vector quantization clustering. Technical report, Texas A&M University, 2006.

[15] Jiali Shen and Housheng Hu. Mobile robot navigation through digital landmarks. *Proceedings of the 10th Chinese Automation and Computing Society Conference in the UK*, pages 117–124, 2004.

[16] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 1998.

[17] Sebastian Thrun, Bucken, Wolfram Burgard, and Dieter Fox. Map learning and high-speed navigation in rhino. *Artificial Intelligence and Mobile Robots*, 1998.

[18] V. Vlahakis, T. Pliakas, A. Demiris, and N. Ioannidis. Design and application of the lifeplus augmented reality system for continuous, context-sensitive guided tours of indoor and outdoor cultural sites and museums. *EG Workshop Proceedings*, 2003.