# IMPLEMENTACIÓN DE UN SISTEMA DE NAVEGACIÓN TOPOLÓGICA PARA UN ROBOT MÓVIL

**FERNANDO BOLIVAR MONSALVE**

**UNIVERSIDAD DEL VALLE**
**FACULTAD DE INGENIERÍA**
**ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**
**PROGRAMA ACADÉMICO DE INGENIERÍA ELCTRÓNICA**
**SANTIAGO DE CALI**
**2012**

# IMPLEMENTACIÓN DE UN SISTEMA DE NAVEGACIÓN TOPOLÓGICA PARA UN ROBOT MÓVIL

**FERNANDO BOLIVAR MONSALVE**

**Trabajo de Grado presentado como requisito parcial para obtener el título de Ingeniero Electrónico**

**DIRECTOR:**
**Ing. EDUARDO CAICEDO Ph.D.**

**CODIRECTOR:**
**Ing. BREYNER POSSO M.Sc.**

**UNIVERSIDAD DEL VALLE**
**FACULTAD DE INGENIERÍA**
**ESCUELA DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**
**ESCUELA DE INGENIERÍA ELÉCTRICA Y ELCTRÓNICA**
**SANTIAGO DE CALI**
**2012**

**Nota de Aceptación**

**Eduardo Caicedo Bravo, Ph.D.**

**Director**

**Breyner Posso Bautista, M.Sc.**

**Codirector**

**Jurado No. 1**

**Jurado No. 2**

# DEDICATORIA

*A mis padres, Susana Monsalve Barrera y Fernando Bolivar Gómez por el apoyo incondicional que me dieron a lo largo de mi carrera, por el desvelo que han tenido conmigo y mis hermanos, por estar en cada etapa de mi vida y comprenderme en los momentos más difíciles. Gracias a ustedes soy una persona íntegra y llena de muchos valores.*

*A mis hermanos, Juan Carlos y Edna Johana por su apoyo incondicional, su paciencia, inteligencia y generosidad.*

*A Claudia Viviana Rubiano, por su compañía, por apoyarme y darme esas ganas de salir adelante y creer en mis capacidades.*

## AGRADECIMIENTOS

El autor agradece a:

De manera especial y sincera al profesor Breyner Posso por aceptar ser mi tutor y darme su apoyo y confianza en mi trabajo y su capacidad para guiar mis ideas ha sido un aporte invaluable, no solamente en el desarrollo de esta tesis, sino también en mi formación profesional. Las ideas propias, siempre enmarcadas en su orientación y rigurosidad, han sido la clave del buen trabajo que hemos realizado juntos, el cual no se puede concebir sin su siempre oportuna participación.

Al Profesor Eduardo Caicedo por haber aceptado ser mi Director en este trabajo de grado y brindarme su orientación y apoyo.

Al grupo de investigación PSI de la Universidad del Valle por haberme facilitado todos los elementos y herramientas para probar el sistema de navegación topológica implementado.

**CONTENIDO**

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# RESUMEN

Se ha implementado un Sistema de Navegación Topológica, que le permite a un robot móvil desplazarse dentro de un entorno estructurado para ir de un punto inicial a otro final. Este sistema está basado en la forma como navegan los seres humanos, es decir, utilizando el reconocimiento de elementos del entorno para lograr llegar de un punto a otro.

Para tal fin se implementó un sistema robótico compuesto por un robot Pioneer 3DX dotado de sonares, encoders, pc-onboard y una cámara que le permite adquirir imágenes del entorno de navegación. Se utilizó un computador donde se ejecutan los módulos de procesamiento de imágenes, reconocimiento de patrones y control del robot y que se comunica con el robot a través de un enlace wi-fi por medio de un router inalámbrico. También se construyeron 31 marcas artificiales donde cada marca son códigos de barras cilíndricas, a las cuales se les diseñó un código de barras de 6 franjas de igual ancho de colores blanco y negro y dos franjas de color magenta ubicadas en los extremos de las franjas en blanco y negro. El color de las franjas blancas y negras de cada marca se intercaló de tal forma que se pudiera diferenciar una marca de otra.

A nivel software el sistema posee 7 elementos básicos. Estos son:

- ***Subsistema de Visión***. encargado de adquirir la información proporcionada por el sensor de visión, procesarla y compararla con una base de datos de puntos de referencia del entorno y enviar los resultados al subsistema deliberativo
- ***Módulo Player.*** Encargado de la comunicación y acceso al hardware del robot Pioneer 3DX, y facilitar la elaboración de aplicaciones propias para el control del robot
- ***Módulo mapa topológico.*** Encargado de localizar al robot dentro del entorno de navegación.
- ***Módulo planificador de misión.*** Tiene la responsabilidad de la planificación de alto nivel, de la cual se encarga el usuario al establecer los parámetros de inicio y destino para la navegación.
- ***Módulo de navegación.*** Encargado de diseñar un camino punto a punto desde la posición actual del robot hasta el punto de meta, en base al mapa topológico que se tiene almacenado a priori.
- ***Módulo piloto.*** Acepta una trayectoria punto a punto desde el módulo de navegación y provee al robot de comportamientos de movimiento que le permitan un desplazamiento seguro
- ***Módulo comportamientos.*** Se encarga de proveer los comportamientos para la ejecución de un movimiento de un punto a otro.

En cuanto al control del robot se implementa un esquema híbrido reactivo/deliberativo distribuyendo la complejidad de la tarea de navegación en dos niveles: en el nivel inferior la capa reactiva que se ejecuta en el robot y se encarga de la evasión de obstáculos en el entorno local y alcanzar metas puestas por el nivel deliberativo; en el nivel superior la capa deliberativa se encarga de crear la ruta para ir de un punto a otro, llevar a cabo la navegación entre estos y relocalizar al robot dentro del mapa topológico.

Es la primera aproximación a la navegación topológica que se desarrolla el sistema implementado en el grupo de investigación Percepción y Sistemas Inteligentes (PSI) con lo cual se abre una nueva línea de investigación en el área de robótica móvil que involucra áreas de trabajos futuros.

***Palabras Claves:*** *arquitecturas de control híbridas, mapa topológico, marca artificial, navegación topológica, Pioneer-3DX, procesamiento de imágenes.*

# ABSTRACT

The Topological Navigation System allows a mobile robot to move within a structured environment to go from a starting point to another end. This system is based on the sail shape as humans, ie, using the recognition of the environment to achieve elements get from one point to another.

The robotic system built has a Pioneer 3DX robot equipped with sonars, encoders, pc-onboard and a camera that allows you to acquire images of the browsing environment. We used a computer where you run the modules image processing, pattern recognition and robot control and communicates with the robot through a Wireless connection via a wireless router. 31 were also constructed in which each mark artificial marks are cylindrical bar codes to which they are designed a bar code 6 strips of equal width of white and black and magenta two strips located at the ends of the strips in black and white. The color of the black and white stripes of each mark is sandwiched in such a way that could distinguish one from another brand.

The system software has 7 basic elements. These are:

- **Vision Subsystem**. Responsible for acquiring the information provided by the vision sensor, process and compare it to a database benchmarks environment and send the results to the deliberative subsystem
- **Player Module**. Responsible for communication and access to Pioneer 3DX robot hardware, and facilitate the development of own applications to control the robot
- Module topological map. Responsible for locating the robot in the navigation environment.
- **Mission planner Module**. Is responsible for the high-level planning, which is responsible for the user to set the parameters for starting and destination for navigation.
- **Navigation Module**. Responsible for designing a way point to point from the current position of the robot to the target point, based on the topological map which is stored a priori.
- **Pilot Module**. Accepts a path from point to point navigation module and provides the robot motion behaviors that allow safe movement.
- **Behaviors Module**. Responsible for providing for the execution behavior of a movement from one point to another.

As for the robot control scheme implements a hybrid reactive/deliberative distributing the complexity of the navigation task into two levels: on the lower level reactive layer that runs on the robot and is responsible for the avoidance of obstacles in the environment local and achieve goals set by the deliberative level, at the top level deliberative layer is responsible for creating the path to get from one point to another, carrying out the navigation between them and relocate the robot in the topological map.

This is the first topological approach to navigation that develops the system implemented in the research group Perception and Intelligent Systems (PSI) thereby opening a new line of research in the area of mobile robotics involves areas of future work.



***Keywords:*** *hybrid control architectures, topological map, artificial mark, topological navigation, Pioneer-3DX, image processing.*

# CAPITULO I

## 1. INTRODUCCIÓN

Siendo la robótica un campo de estudio nacido años atrás, cuenta hoy en día con un historial considerable en cuanto a trabajos de investigación, desarrollos y propuestas que en gran medida sustentan las bases para pruebas de mayor alcance e inclusive nuevas propuestas o líneas de investigación.

Para la realización del presente trabajo es necesario conocer antes distintos algoritmos enfocados a tres principales problemas como lo son: la planeación de trayectorias enfocadas a la exploración de un ambiente, la correcta detección de obstáculos y técnicas para evitar colisiones, así como la determinación de posición y orientación del robot en el espacio. Para los tres diferentes problemas, existen una variedad de algoritmos por lo que el primer objetivo es conocer algunos de ellos para poder seleccionar los más adecuados para el tipo de aplicación que se vaya a implementar.

Para hablar de una solución eficiente no se debe olvidar que ésta no solo depende de la tarea de exploración sino que al mismo tiempo depende de las técnicas para la adquisición de información y su procesamiento, el método para la identificación de los objetos en el ambiente y finalmente la representación gráfica del espacio explorado. Por ello es importante llevar a cabo una revisión de las diferentes técnicas y algoritmos existentes para finalmente poder elaborar un algoritmo para la resolución del problema y que mejor se pueda adaptar a las necesidades y características del equipo disponible, ya que el acercamiento que cada trabajo tiene hacía un mismo problema varía de acuerdo a los recursos disponibles haciendo que un algoritmo pueda ser mucho más eficaz con un tipo de dispositivos que con otro.

Los algoritmos difieren uno de otro en aspectos tan simples como que para algunos el cubrir varias veces un mismo lugar de la zona explorada representa un costo inútil en tiempo, mientras que bajo otras circunstancias como un ambiente en continuo cambio, estas repetidas revisiones de un mismo lugar representan una ventaja ya que mantienen actualizada la información y con ello un conocimiento real del terreno. Para un robot inmerso en un ambiente desconocido, los algoritmos para planeación de trayectorias no resultan ser tan adecuados para la exploración, pues en general dan por hecho el conocimiento de la zona sobre la que se va a mover el robot, es decir, los algoritmos para planeación de trayectorias toman como entrada un mapa sobre el cual debe navegar el robot y entregan como salida una secuencia de movimientos que se encadenan como una trayectoria para el robot buscando generalmente llevarlo del punto actual a un punto diferente definido como meta. Ahora bien, en la robótica muchas veces se trabaja realizando solamente simulaciones con los algoritmos propuestos ejecutándolos exclusivamente en la computadora, sin embargo las cosas son muy diferentes cuando se pasa de una simulación a un modelo real en el cual las consideraciones deben ampliarse y se deben dejar de lado los casos ideales asumidos.

Un problema cuando se trabaja con un robot dentro de un espacio real es ocasionado por los sensores cuando se desea hacer la generación y actualización del mapa de la zona explorada ya que los sensores son dispositivos muy propensos a ruido y suelen entregar datos imprecisos y, debido a que los sensores son nuestro medio para de alguna manera percibir el ambiente en el cual actúa el robot, muchas veces los datos de los sensores arrojan una apreciación errónea.

Los datos que el robot capta del ambiente a través de sus sensores sirven obviamente para la navegación, sin embargo también las lecturas de los sensores son empleadas para estimar la posición del robot en el ambiente, lo que implica entonces que la determinación precisa de dicha posición sea también un problema implícito, y esto hace surgir también diferentes técnicas de ubicación para el robot dentro del espacio.

En general a un robot móvil autónomo se le requieren ciertas capacidades como lo son la utilización de diversos sensores, adquirir y manipular un modelo de su entorno operacional mediante la extracción y la interpretación del mundo real. Sin embargo, los diferentes sensores están sujetos a limitaciones intrínsecas que solo pueden ser resueltas por la utilización de sistemas multisensoriales. Asimismo, los robots móviles recorren grandes áreas y deben combinar vistas obtenidas desde muchas localizaciones diferentes enfocadas hacia un modelo único y consistente del entorno, que refleje la información obtenida.

Una de las formas de aunar dicha información es considerar información topológica, frente a información métrica. Mientras la información geométrica plantea el problema de la imprecisión asociada a los sensores, los mapas topológicos representan la información mediante características del entorno y la relación existente entre ellas. Dicho de otra manera, mientras que con los mapas geométricos necesitamos información métrica para podemos relacionar con el entorno, con los mapas topológicos sólo requerimos relaciones entre los elementos del entorno. Esta última forma de navegar, es la forma en la que navegamos los humanos.

Los mapas topológicos surgen por los estudios de algunos autores como Kuipers **[KUIPERS 87]**. Otros autores **[NEHM95] [TRULLIER97]** han estudiado la forma de representar el entorno y de moverse en él realizada por los animales y los humanos, llegando a la conclusión de que la forma natural de representar el entorno es mediante estructuras y relaciones topológicas.

## 1.1 PLANTEAMIENTO DEL PROBLEMA

La Robótica Móvil se entiende como la disciplina que investiga e intenta crear sistemas con la capacidad de desplazarse y realizar algún tipo de tarea en el entorno en el que se encuentran. En general se busca que dicho desplazamiento pueda ser autónomo, es decir, que no dependa de la intervención de algún operador humano. Para lograr este objetivo se crean sistemas de navegación para los robots móviles.

La navegación, en el contexto de la robótica, se define como la metodología que permite movimiento de un robot móvil a través de un entorno evitando los obstáculos. El problema de la navegación de un Robot Móvil plantea tres tareas: percibir el entorno a través de los sensores del robot, con lo cual éste crea una representación del mundo que lo rodea; construir una trayectoria libre de obstáculos, para que el Robot llegue a su punto de destino; y guiar el Robot a través de la trayectoria construida. **[MARTÍNEZ 97]**

A finales de los años 80 comienzan a aparecer los primeros sistemas que intentan construir o actualizar un mapa del entorno mientras que el Robot se desplaza de forma autónoma. Los algoritmos utilizados tratan de darle solución al problema de la construcción del mapa y de la localización del robot, de tal forma que construyen el mapa con la suposición de que la posición del Robot es correcta y se localiza en el mapa suponiendo que este es correcto. Pero con el tiempo se dan cuenta que la solución al problema no es posible sin considerar ambos aspectos simultáneamente.

Desde la aparición del artículo introductorio de Smith, Self y Cheeseman en 1988 **[SMITH 88]** el problema de la construcción de un mapa y localización simultanea se ha venido denominando en la literatura como el problema SLAM (del inglés Simultaneous Localization and Mapping), aunque también es conocido en menor medida como el problema CML (del inglés Concurrent Mapping and Localization).

Este problema (navegación de un Robot Móvil) no ha logrado solucionarse totalmente y continúa siendo el tema de investigación principal de grupos de robótica en todo el mundo, a pesar de que se han llevado a cabo considerables avances en el área.

Los métodos tradicionales de navegación de robots móviles, basados sólo en información geométrica, plantean grandes dificultades al intentar reducir simultáneamente la incertidumbre que se tiene en el conocimiento de la posición del robot y de su entorno.

Actualmente la navegación topológica está considerada como la forma más simple y natural de navegación autónoma en robots móviles. La navegación topológica utiliza unos puntos de referencia del entorno en el que se trabaja.

Un ejemplo de este tipo de navegación es la capacidad de enviar a un robot un comando del tipo “sigue el pasillo y al final gira hacia la izquierda y entra en la tercera habitación de la derecha”. Incluso sin un mapa, el robot tiene suficiente información como para guiarse, a través de los pasillos indicados sin perderse, siguiendo un conjunto de marcas que podrían ser: “pasillo”, “final”, “tercera habitación”.

Existen dos tipos de navegación topológica: la navegación topológica relacional que es la técnica más utilizada y se basa en una serie de mapas en los cuales los puntos claves son los nodos de un grafo relacional y el camino para llegar de un nodo a otro son las aristas de dicho grafo; y la navegación topológica asociativa que al contrario que la técnica relacional, se basa fundamentalmente en crear un comportamiento que traduce un estímulo observado en una dirección de movimiento para alcanzar el patrón o marca observado. **[MUÑOZ 99]**

En el Grupo de Investigación Percepción y Sistemas Inteligentes (P.S.I) de la Universidad del Valle no se han llevado a cabo propuestas de Navegación Topológica en ambientes estructurados y semi-estructurados por lo que surge la pregunta ¿Es posible desarrollar estrategias de Navegación Topológica en la plataforma del Robot Móvil Pioneer 3DX y qué tipo de sensores serían los más adecuados y cuáles serían las estrategias a implementar.

## 1.2 OBJETIVOS DEL PROYECTO

### 1.2.1 Objetivo General

Implementar un sistema de navegación topológica que le permita a un robot móvil desplazarse en un entorno estructurado.

### 1.2.2 Objetivos Específicos

- Estado del Arte de los diferentes métodos utilizados para implementar la navegación topológica y seleccionar uno.

- Estado del Arte de las técnicas de procesamiento de imágenes más utilizadas en la navegación topológica e implementar una.

- Implementar un sistema de navegación topológica en la plataforma robótica Pioneer 3Dx.

- Establecer la tarea de navegación y el procedimiento de validación del sistema.

## 1.3 JUSTIFICACIÓN

La navegación autónoma representa uno de los mayores retos a conseguir dentro de la robótica, ya que involucra prácticamente todos los campos de la inteligencia artificial y robótica: sensado, actuación, planificación, arquitecturas de control, hardware, eficiencia computacional y resolución de problemas.

Los esquemas de navegación pueden ser puramente reactivos, deliberativos o híbridos (una combinación de los dos anteriores). El primer tipo provee un buen método para desplazarse por el entorno evadiendo los obstáculos, sin embargo es el segundo método el que le permite al robot generar un plan para llegar a un lugar determinado. El tercer tipo permite combinar las ventajas de los sistemas reactivos y deliberativos.

Uno de los esquemas deliberativos más utilizados para la navegación de los robots móviles es la navegación topológica debido a que es más simple y natural pues se basa en una representación topológica del entorno donde el robot lleva a cabo su tarea.

La navegación autónoma es uno de los temas de investigación en robótica móvil que más interés ha despertado en diversas universidades alrededor del mundo y la Universidad del Valle no es la excepción. Con este proyecto se pretende contribuir al desarrollo de la línea de investigación en robótica móvil del grupo de investigación Percepción y Sistemas Inteligentes (PSI) fortaleciendo el campo de los sistemas de navegación deliberativos.

## 1.4 ORGANIZACIÓN DEL DOCUMENTO

Este documento se compone de cinco capítulos, referencias bibliográficas y anexos que sustentan el desarrollo del proyecto.

El capítulo 2, presenta un estado del arte del área de conocimiento sobre el cual se enmarca este proyecto. Se muestran las estrategias de navegación para robots móviles, los diferentes métodos utilizados para implementar la navegación topológica,

las técnicas de procesamiento de imágenes más utilizadas en este tipo de navegación, y las arquitecturas de control.

El capitulo 3, presenta los elementos y herramientas que permiten el desarrollo del proyecto, describiendo la plataforma de experimentación, las herramientas software, el sensor de visión del Robot Móvil y el entorno de experimentación.

El capitulo 4, muestra el desarrollo del sistema de navegación topológica, presentando la implementación de los módulos que componen el sistema y el funcionamiento de la aplicación final desarrollada.

El capítulo 5, describe las pruebas y los resultados obtenidos, identificando los alcances y desempeño de cada uno de los módulos descritos en el capitulo anterior, validando su funcionalidad e identificando las limitaciones del sistema.

El capítulo 6, presenta las conclusiones del proyecto, además de algunas recomendaciones.

El capítulo 7, define los trabajos futuros para el sistema de navegación topológica implementado.

# CAPITULO II

## 2. ESTADO DEL ARTE

### 2.1 NAVEGACIÓN [MUÑOZ 99]

En robótica, la navegación hace referencia a la metodología que permite guiar el movimiento de un robot a través de un entorno para llegar a un destino, evadiendo los obstáculos que encuentre en su camino.

La navegación como tal es una de las tareas que más desafíos plantea ya que involucra muchos subproblemas, tales como: sensado, planificación, actuación y arquitecturas de control.

Para llevar a cabo esta tarea se necesitan funciones de navegación que permitan contestar las siguientes preguntas:

- **¿Dónde voy?** Esta pregunta generalmente es contestada por el diseñador del robot o por la naturaleza de la tarea a llevar a cabo.

- **¿Cuál es el mejor camino para llegar al destino fijado?** En este punto entra en juego la problemática de la planificación de rutas y la creación de mapas de forma autónoma por parte del robot.

- **¿Dónde he estado?** El propio robot necesita conocer su recorrido para, si no ha estado nunca en ese lugar, crear un mapa asociado al existente, o si ya ha estado, conocer su situación en dicho mapa o añadir cambios que pudieran haber sucedido en el entorno (paredes nuevas, muebles cambiados de lugar, etc.).

- **¿Dónde estoy?** Esta cuestión está estrechamente relacionada con la anterior, puesto que debe existir un sistema de localización, tanto actual como pasada, para conocer la posición del robot en el mapa creado en un principio. Esta localización puede ser relativa a un entorno local en coordenadas topológicas (el robot se encuentra en el centro de una determinada habitación) o en coordenadas absolutas (latitud, longitud y altitud).

Un ejemplo de lo descrito pueden ser los robots guías en museos, como lo son el robot Minerva de la Carnegie Mellon University (Figura 1) y el robot Rhino de la Universidad de Bonn (Figura 2). Estos dos robots son capaces de llevar a cabo todas las funciones de un guía humano en un museo, incluida la de guiar al grupo a determinadas salas bajo demanda de alguno de los visitantes.

Figura 1. Robot guía Minerva **[Fotografía cortesía del Instituto de robótica de la universidad Carnegie Mellon]**

Figura 2. Robot guía Rhino en el Museo de Bonn **[Fotografía cortesía del Museo Bonn]**

### 2.1.1 Navegación en Interiores

Algunos de los primeros sistemas de visión desarrollados para navegación de robots se basaban en gran medida en la geometría del espacio y otra información métrica para llevar a cabo el proceso de visión y la realización de la auto-localización, para ello el espacio interior era representado por modelos CAD, aunque estos modelos fueron reemplazados por modelos más simples como mapa de ocupación, mapas topológicos o inclusive secuencias de imágenes.

La clasificación de los sistemas de navegación según DeSouza y Kak [**DESUOZA 02**] se divide en tres:

- Navegación basada en mapas.
- Navegación basada en construcción de mapas.
- Navegación sin mapas.

#### *2.1.1.1 Navegación basada en mapas*

La idea central en cualquier navegación basada en mapas es proveer al robot, directa o indirectamente, de una secuencia de puntos de referencia que se espera sean encontrados durante la navegación, la tarea del sistema de visión es entonces buscar e identificar los puntos de referencia observados en una imagen. Una vez que éstos son identificados, el robot puede usar el mapa provisto para realizar una auto-

localización. Los cálculos involucrados en la localización basada en visión pueden ser divididos en los siguientes cuatro pasos:

- Adquirir información sensorial: adquirir y digitalizar imágenes.

- Detectar puntos de referencia: extraer bordes, suavizar, filtrar y segmentar regiones en base a diferencias en sus niveles de gris, color, profundidad o movimiento.

- Establecer relaciones entre la observación y la expectativa: el sistema trata de identificar los puntos de referencia observados buscando en la base de datos posibles coincidencias de acuerdo a un criterio de medición.

- Calcular la posición: una vez que una coincidencia o un conjunto de coincidencias es obtenido, el sistema necesita calcular su posición como una función de los puntos de referencia observados y sus posiciones en la base de datos.

#### *2.1.1.2 Navegación basada en construcción de mapas*

El fundamento de este enfoque es que no siempre es fácil generar descripciones de un modelo, especialmente cuando se tiene que proveer información métrica, por lo cual se ha propuesto que robots autónomos o semiautónomos pudieran explorar su ambiente y construir una representación interna de éste, cabe resaltar que este enfoque propone el uso de otro tipo de sensores y no usa solamente sensores de visión.

#### *2.1.1.3 Navegación sin mapas*

Esta categoría incluye a todos los sistemas en los cuales la navegación es llevada a cabo sin una previa descripción del ambiente y sin la creación de un mapa. Los movimientos del robot son determinados observando y extrayendo información relevante de los elementos en el ambiente, estos elementos pueden ser paredes, escritorios, puertas, etc., además de que tampoco es necesaria que las posiciones de estos elementos dentro del ambiente sean conocidas.

### 2.1.2 Esquemas de navegación en robots móviles [MUÑOZ 99]

Realizar una tarea de navegación para un robot móvil significa recorrer un camino que lo conduzca desde una posición inicial hasta otra final, pasando por ciertas posiciones intermedias o submetas. El problema de la navegación se divide en las siguientes cuatro etapas:

- **Etapa I. Percepción del mundo:** Mediante el uso de sensores externos (láser, sonares, cámaras, etc.), se crea un mapa o modelo del entorno donde se desarrollará la tarea de navegación.

- **Etapa II. Planificación de la ruta:** Se crea una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el robot móvil. Esta secuencia se calcula utilizando el modelo o mapa de entorno, la descripción de la tarea que debe realizar y algún tipo de procedimiento estratégico.

- **Etapa III. Generación del camino:** En primer lugar se define una función continua que interpola la secuencia de objetivos construida por el planificador. Posteriormente se procede a la discretización de la misma a fin de generar el camino.

- **Etapa IV. Seguimiento del camino:** Se efectúa el desplazamiento del Robot Móvil, según el camino generado mediante el adecuado control de los actuadores del vehículo.

La complejidad del sistema necesario para desarrollar la planificación de la ruta depende principalmente del conocimiento que se posee del entorno de trabajo. Si el modelo del entorno del que dispone el robot no posee la información suficiente para la construcción de un camino libre de obstáculos, éste será ineficaz.

Por lo tanto se necesitará dividir la tarea de planificación en dos subtareas: planificación global y planificación local.

- **Planificación global:** construir o planificar la ruta que lleve al robot a cada una de las submetas determinadas, según las especificaciones del problema que debe resolverse. Esta planificación es una aproximación al camino final que se va a seguir.

- **Planificación local:** consiste en resolver las obstrucciones sobre la ruta global en su entorno local para determinar la ruta real que será seguida. El modelo del entorno local se construye mediante la fusión de la información proporcionada por los sensores externos del robot móvil.

La construcción de la ruta global puede realizarse antes de que el vehículo comience a ejecutar la tarea, mientras que la planificación local se lleva a cabo en tiempo de ejecución.

En el caso de realizar una navegación sobre entornos totalmente conocidos resulta innecesario proceder a una planificación local, pero a medida que disminuye el conocimiento de la zona por la cual el robot móvil realiza su tarea, aumenta la relevancia de la misma.

En aplicaciones de navegación en exteriores, el conocimiento que se posee del entorno es pobre y por tanto se necesita hacer un uso intensivo de la planificación local para construir de forma dinámica el camino que debe seguir a medida que se navega. En este esquema se recurre a un uso constante del sistema sensorial y el planificador local se responsabiliza de la coordinación de la percepción, planificación y control del vehículo para guiarlo por el camino especificado, mientras verifica el entorno y realiza el sorteo de obstáculos.

La diferencia entre los esquemas de navegación presentados radica en la adaptación que debe poseer el robot móvil para moverse por su entorno de trabajo según el conocimiento de que disponga sobre la estructura del mismo, dándose mayor ponderación a la planificación global o local. Sin embargo, mantienen un nexo común en la realización de forma secuencial y continua las operaciones de percepción, planificación, generación y seguimiento de trayectorias.

Este grupo de operaciones permite el desarrollo de la navegación minimizando cierto índice de coste, como puede ser la distancia recorrida o el consumo energético del vehículo. Se habla en este caso de una descomposición jerárquica en módulos funcionales que encadenados en forma de ciclo realizan la navegación estratégica.

Este tipo de navegación se caracteriza por la necesidad de conocer con el mínimo error posible la posición actual del Robot Móvil, ya que la realimentación de esta información es la base para el cálculo de la próxima acción de control. Mediante el uso de la odometría del Robot Móvil se puede estimar su posición, pero debido a la naturaleza del método y a las características de los sensores utilizados, la estimación efectuada se ve afectada por errores acumulativos. Cuando dichos errores alcanzan niveles indeseables se hace necesario eliminarlos mediante la utilización de algoritmos de estimación de la posición basados en referencias externas.

El problema de estos errores es que pueden aparecer inesperadamente (por ejemplo cuando el robot pasa por encima de un objeto que se encuentra en el suelo) y pueden causar errores muy grandes en la estimación de la posición.

Se han desarrollado algoritmos para estimar la incertidumbre de la posición de un robot que utiliza odometría. Según esto, cada posición calculada por el robot está rodeada por una *elipse de error* característica, la cual indica la región de incertidumbre para la posición actual del robot. Estas elipses crecen a medida que la distancia recorrida aumenta, a no ser que un sistema de estimación de la posición absoluto reduzca el crecimiento de dicha incertidumbre, como se muestra en la Figura 3.

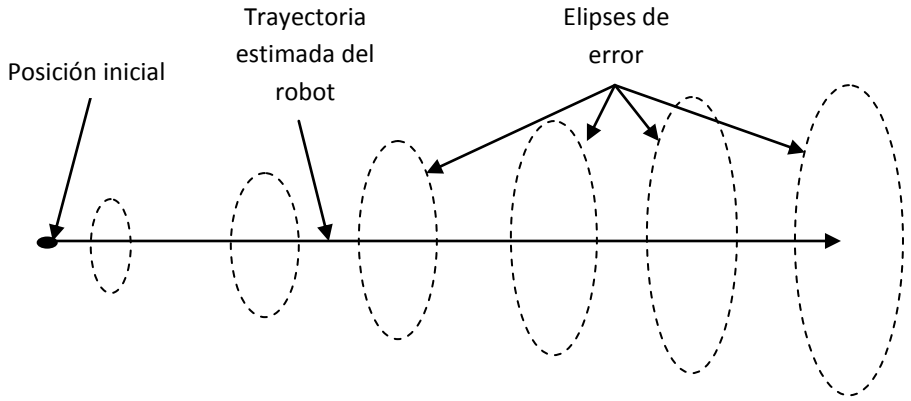



Figura 3. Elipses de error estimadas en torno a la trayectoria de un robot móvil

La navegación estratégica tiene sus limitaciones en entornos dinámicos no conocidos, ya que requiere un completo conocimiento de la dinámica de los posibles obstáculos móviles, además de una adecuada actualización del mapa de entorno.

La filosofía alternativa a la navegación estratégica consiste en el uso intensivo de sensores de bajo coste con el fin de reaccionar dinámicamente ante el entorno, con lo cual pierde relevancia el concepto de planificación y seguimiento de caminos. Esto se basa en una arquitectura descompuesta en módulos especializados en realizar tareas individuales o comportamientos. Esta alternativa se comporta de forma eficiente en entornos dinámicos donde se posee un conocimiento impreciso del mismo.

En cada intervalo de navegación, el sistema sensorial, según la información extraída del entorno local del robot, activa uno o varios comportamientos simples que suman

sus actuaciones, con lo que el comportamiento final resulta una mezcla de los simples activados (Figura 4).

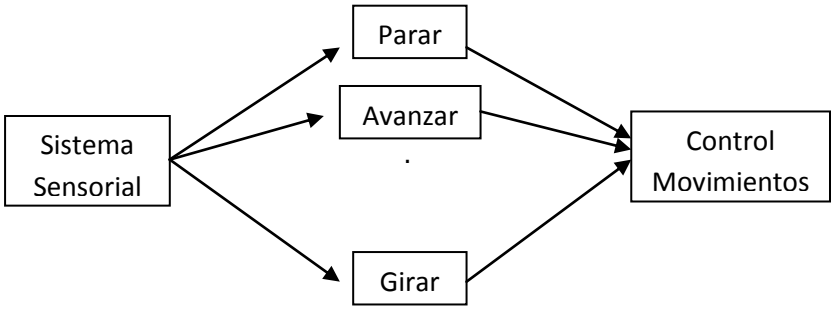



Figura 4. Navegación reactiva basada en comportamientos

La navegación reactiva basada en comportamientos ha sido implementada en múltiples aplicaciones entre las que predominan los comportamientos de supervivencia, dando lugar a robots errantes que se mueven con libertad por entornos desconocidos e incluso dinámicos, sin colisionar con los obstáculos, pero que raramente obedecen a un plan establecido, imprescindible en misiones reales.

## 2.2 ARQUITECTURAS PARA CONTROL DE ROBOTS

Existen diferentes tipos de arquitecturas diseñadas teniendo en cuenta especificaciones sobre eficiencia, tiempo de respuesta ante acontecimientos no previstos y disponibilidad de información del entorno. Se destacan el control reactivo, el deliberativo y una arquitectura híbrida reactiva-deliberativa **[OLLERO 01].**

El ***control reactivo*** es una técnica basada en una asociación estrecha entre los estímulos obtenidos de los sensores, y las acciones disparadas sobre los efectores. Este tipo de control permite responder de manera muy veloz a entornos cambiantes y sin una estructura definida. Las limitaciones de este enfoque son que este tipo de robots, al solo buscar acciones para un estímulo dado, no suelen almacenar información, no tienen memoria, ni representaciones internas del entorno, ni habilidad de aprender con el tiempo.

En el ***control deliberativo*** el robot toma toda la información sensorial disponible y todo el conocimiento que tiene almacenado internamente, y razona con base en ello para crear un plan de acción. Para esto, el robot debe realizar una búsqueda sobre (potencialmente) todos los planes posibles hasta encontrar uno que sea útil. Eso puede llevar mucho tiempo, razón por la cual si el robot debe reaccionar velozmente, puede no ser práctico.

En ocasiones se utilizan **arquitecturas híbridas** que ofrecen un compromiso entre las puramente reactivas y las deliberativas. Se emplea un sistema reactivo para control de bajo nivel y un planificador para la toma de decisiones de un nivel superior. Se separa el sistema de control en dos o más partes comunicadas pero básicamente independientes.

En la mayor parte de los casos el proceso reactivo de bajo nivel se encarga de funciones de seguridad, mientras que el de alto nivel utiliza un planificador para seleccionar la acción.

En la Figura 5 se muestra un esquema de los tres niveles que posee este tipo de arquitectura: el componente reactivo (basado en comportamientos) en el nivel inferior, el componente deliberativo en el nivel superior y un mecanismo de coordinación en el nivel intermedio.

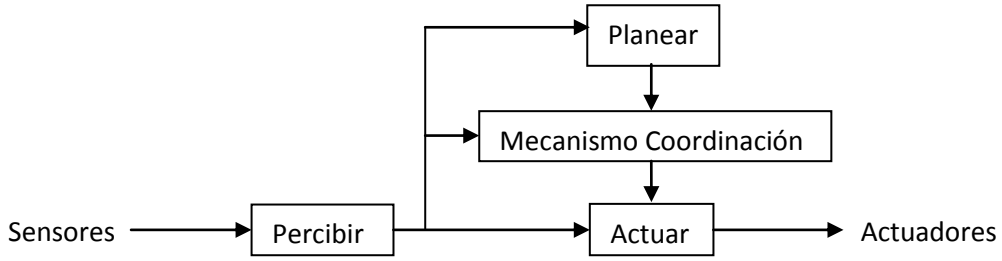



Figura 5. Esquema general de una arquitectura de control híbrida.

Algunos ejemplos de este tipo de arquitectura son AURA **[ARKIN 92]**, SFX **[MURPHY 00],** HILARE **[ALAMI 00].**

La principal aportación de la arquitectura **AuRA** (Autonomous Robot Architecture) está en que utiliza una navegación basada en campos de potencial, de carácter inminentemente reactivo. Sin embargo, va a ser considerada como una arquitectura híbrida ya que presenta módulos de planificación.

Arkin **[ARKIN 92]** incorpora, en cambio, el concepto de múltiples comportamientos concurrentes que él denomina “esquemas” y que producen acciones del mismo tipo que finalmente se combinan utilizando una metodología de campos de potenciales.

Arkin define la arquitectura **AuRA** para un robot autónomo **[ARKIN 89]** siguiendo el modelo de navegación animal propuesto por Arbib y House para explicar el comportamiento de las ranas. **AuRA** tiene las siguientes propiedades:

- Utiliza un régimen de control motriz o esquemas motrices como mecanismos de comportamientos reflexivos o percepción orientada a la acción.
- No se utiliza un conocimiento de alto nivel para el entorno o para la selección de estrategias motrices y de percepción.
- No hay una planificación a priori, en su lugar se activa un esquema del tipo ‘mueve hacia objetivo’ o ‘mueve hacia adelante’.
- No existen capas de esquemas como en el modelo de Brooks.
- Los esquemas son activados dinámicamente mediante mecanismos basados en percepción según el robot avanza a través del mundo externo.
- Cada vector de salida de los esquemas motrices que fueron activados se añade y normaliza para conseguir la velocidad y dirección en las cuales el robot debería moverse.

Esta arquitectura ha sido desarrollada por R.C Arkin **[ARKIN 92]** en la Universidad de Massachusetts. En su desarrollo este investigador introduce el concepto de varios comportamientos elementales que él denomina esquemas (Un término derivado de las ciencias biológicas).

Arkin, ha implementado esta arquitectura para el control de un robot móvil autónomo. En este caso los comportamientos elementales que mencionan son por ejemplo: “evitar objetos estáticos”, “moverse hacia adelante”, “permanecer en un camino”, etc. Todos estos comportamientos están pensados para realizar tareas de tipo navegacional y se implementan como procedimientos parametrizados.

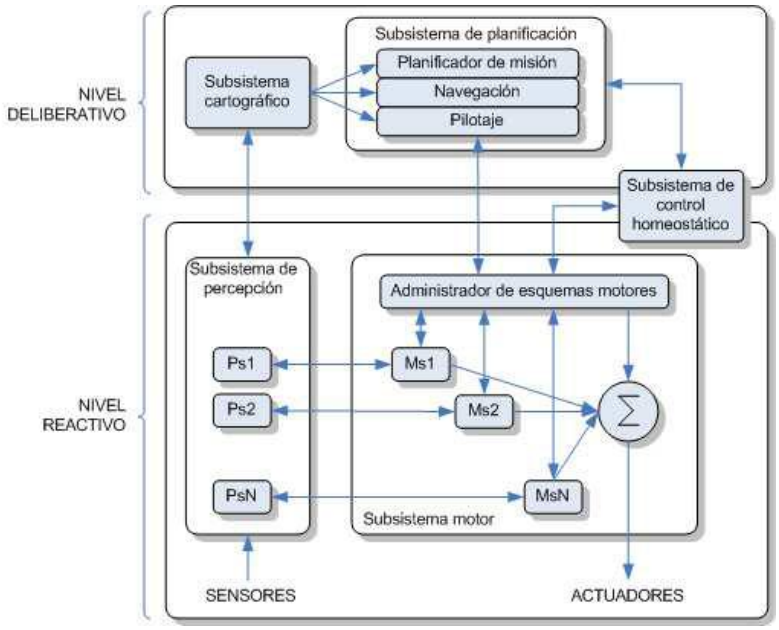


Figura 6. Arquitectura de control AuRA para la navegación de robots.

***Módulo Planificador de Misiones:*** El planificador de misiones tiene la responsabilidad de la planificación de alto nivel. Este incluye capacidad de razonamiento espacial, determinación de los parámetros necesarios para el módulo de navegación y de pilotaje, así como modos de operación y criterios de optimización. Los comandos de misión los introduce un operador a través de una interface de usuario. La salida del módulo de planificación se dirige hacia el módulo de navegación y consiste en modos de operación, que determinan el comportamiento global del robot así como una lista de parámetros, adicionalmente se proveen submetas para la tarea que se va a realizar.

***Módulo de Navegación:*** Este módulo diseña un camino punto a punto desde la posición actual del robot hasta el punto de meta, en base a un mapa del entorno que se tiene almacenado a priori. El nivel de representación usado por este módulo es un modelo de grafo de espacios libres donde los obstáculos se aproximan como polígonos. La trayectoria se calcula en base a la ubicación de los vértices de éstos.

Constantemente el módulo de navegación reporta al planificador de misión el estado de la misma. Se representa el entorno inmediato al vehículo, donde se mantiene información acerca de las marcas visibles, características de los obstáculos conocidos, atributos del terreno, etc. Estos datos están disponibles para la predicción del subsistema de percepción o para ser usados por el módulo de pilotaje en la selección y ejecución de comportamientos.

La salida de este módulo se dirige al subsistema piloto, ésta consiste en una trayectoria punto a punto así como algunos parámetros que afectan el comportamiento de este bloque.

Esencialmente el módulo de navegación se basa en un modelo mientras que el piloto se basa en la información sensorial. El sistema piloto reporta constantemente al módulo superior el cumplimiento o no de las submetas establecidas. Si ha existido un fallo del piloto, el módulo de navegación puede iniciar una replanificación sin invocar al planificador de misión.

***Módulo de Pilotaje:*** El módulo de pilotaje acepta una trayectoria punto a punto desde el módulo de navegación y provee al robot de comportamientos de movimiento que le permitan un desplazamiento seguro.

Este módulo selecciona los comportamientos (esquemas) apropiados de un repertorio de ellos y se los envía parametrizados hacia el módulo que se encarga de instanciarlos.

A partir de este punto se ejecuta un ciclo a través del "administrador de esquemas". Durante el movimiento, el módulo cartógrafo construye una representación temporal del entorno a partir de los datos provenientes de los sensores. Si por algún motivo el administrador de comportamientos falla en alcanzar un punto durante el período de tiempo determinado entonces se reinvoca al piloto para encontrar un camino alternativo en base a los mapas a priori y temporal. Polígonos aproximados representando objetos detectados se insertan en el mapa y se corre otra vez el algoritmo de búsqueda de espacio libre.

Los comportamientos de movimiento con los que trabaja Arkin son:

- Moverse en una dirección
- Moverse a un punto.
- Evitar obstáculos estáticos.
- Parada de emergencia.
- Mantenerse en una trayectoria identificable (camino, sendero, etc.).

Se incluyen también comportamientos de percepción como por ejemplo:

- Encontrar un obstáculo.
- Encontrar una referencia.
- Encontrar un camino.

A fin de proveer una acción de movimiento apropiada para recorrer una trayectoria, el módulo piloto accede a la información que se encuentra presente tanto en el mapa a priori como en el mapa temporal, que representa los alrededores del robot y conjuntamente con las submetas provistas por el módulo de navegación las almacena en una base de hechos al inicio de cada etapa.

Arkin utiliza un sistema de reglas para seleccionar los comportamientos en base a las submetas de navegación. El piloto mantiene también una base de reglas que se aplican al contexto actual (base de hechos). El resultado es una lista de comportamientos movimiento y de percepción parametrizados y optimizados para una particular etapa navegacional. Algunos comportamientos se instancian siempre como el de evitar obstáculos estáticos mientras otros se ejecutan de acuerdo al contexto.

***Administrador de Comportamientos:*** El control distribuido para la ejecución de un movimiento de un punto a otro ocurre dentro del módulo administrador de comportamientos. Múltiples comportamientos están activos durante el movimiento del robot a fin de lograr una correcta transición de la trayectoria. Arkin usa una metodología de campo de potencial para calcular los comandos de velocidad y curvatura que se darán a los actuadores del robot. Se calcula un vector de velocidad total a partir de las contribuciones vectoriales individuales de cada comportamiento.
Cuando el robot se mueve, el sistema cartógrafo, utilizando los datos sensoriales construye un modelo del mundo percibido en la zona de memoria temporal. Si la trayectoria actual del robot se desvía demasiado de la especificada inicialmente por el módulo de navegación debido a la presencia de obstáculos o errores posicionales se vuelve a invocar al módulo de navegación y se calcula una trayectoria global diferente. Si la trayectoria del vehículo está dentro de límites aceptables (lo que se determina en los módulos superiores) el subsistema de pilotaje y el administrador de comportamientos intentan evitar el obstáculo, seguir un camino o afrontar otros problemas.

Desarrollada por Robin Murphy **[MURPHY 00],** la arquitectura **SFX** (Sensor Fusion Effects) puede considerarse una extensión de la arquitectura **AuRA**, donde se añade una capa de robustez con la incorporación de procesos de fusión sensorial y la inclusión de la tolerancia a fallos (Figura 7). La información sensorial está disponible tanto en el nivel reactivo como en el deliberativo mediante el uso de estructuras de datos tipo pizarra **[NII 89]**, comunes en numerosos sistemas de Inteligencia Artificial para formar estructuras de conocimiento global **[ARKIN 92]**.

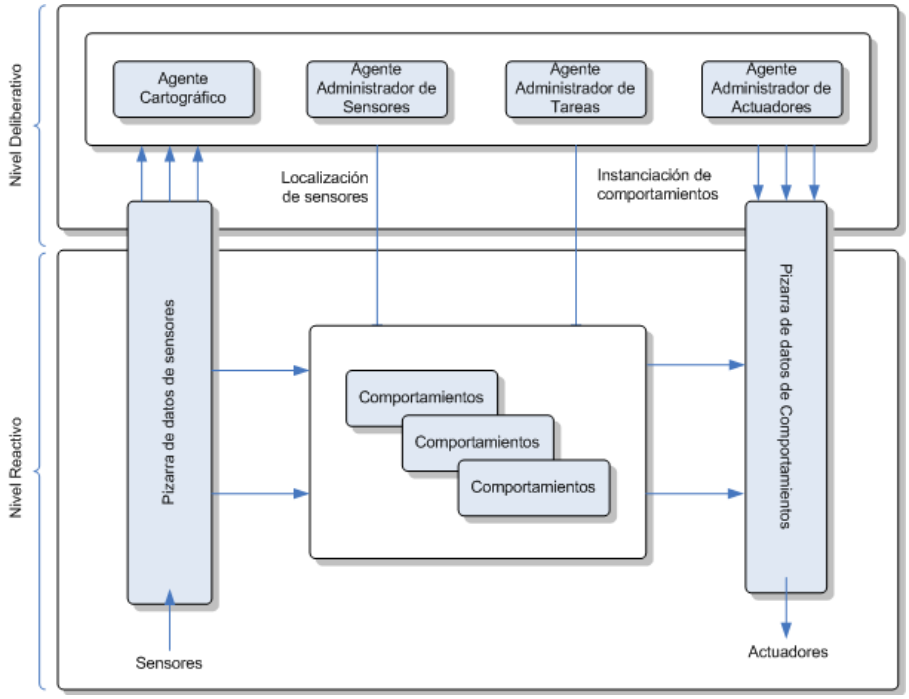


Figura 7. Arquitectura de control de robots SFX.

El nivel deliberativo está formado por componentes software independientes denominados agentes que están especializados en determinadas funciones y pueden interactuar entre ellos. El agente principal es el Planificador de misiones, interactúa con el usuario y especifica las directrices de la misión a los otros agentes del nivel

deliberativo. El objetivo de los agentes es encontrar los comportamientos necesarios que aseguren la realización de la misión cumpliendo con las restricciones establecidas. Asimismo, existen agentes para determinar los sensores que utilizarán los esquemas de percepción y esquemas motores de los distintos comportamientos. Si un sensor fallara, estos agentes podrían averiguar la causa del fallo e intentar solucionarlo o podrían decidir el uso de algún sensor alternativo mediante la incorporación, en el comportamiento, del esquema de percepción correspondiente.
El nivel reactivo se divide en dos capas identificadas por los comportamientos estratégicos y por los comportamientos tácticos. A diferencia de la arquitectura **AuRA**, donde se utilizan campos potenciales para combinar los comportamientos, **SFX** utiliza un método de filtrado donde algunos comportamientos fundamentales inhiben a otros comportamientos.

La idea es similar al proceso realizado en la arquitectura reactiva Subsumption de Brooks **[BROOKS 86]** pero en este caso los comportamientos de la capa inferior (comportamientos tácticos) inhiben a los comportamientos de la capa superior (comportamientos estratégicos). Por ejemplo, la velocidad emergente en la arquitectura **AuRA** es siempre el resultado de la suma ponderada de todos los comportamientos activos, mientras que en la arquitectura **SFX** existe un control de la velocidad máxima a través de un comportamiento táctico específico. Esto quiere decir que los comportamientos estratégicos de la capa superior proporcionan una velocidad al igual que ocurre en la arquitectura **AuRA** y, después, el comportamiento táctico de la capa inferior filtra este valor dependiendo de si es superior al valor máximo o no. La velocidad resultante siempre será el valor menor de ambos. **SFX** basa esta técnica en la idea de que la mayoría de las tareas pueden expresarse mediante un solo comportamiento estratégico y uno o varios tácticos que actúen de filtros del estratégico.

La arquitectura **HILARE**, desarrollada en el **LAAS** (Laboratoire d'analyse et d'Architecture des Systèmes, Toulouse) y especificada **[ALAMI ET AL. 00]**, se compone de tres niveles jerárquicos, con diferentes restricciones temporales y con manipulación de una distinta representación de datos (Figura 8).

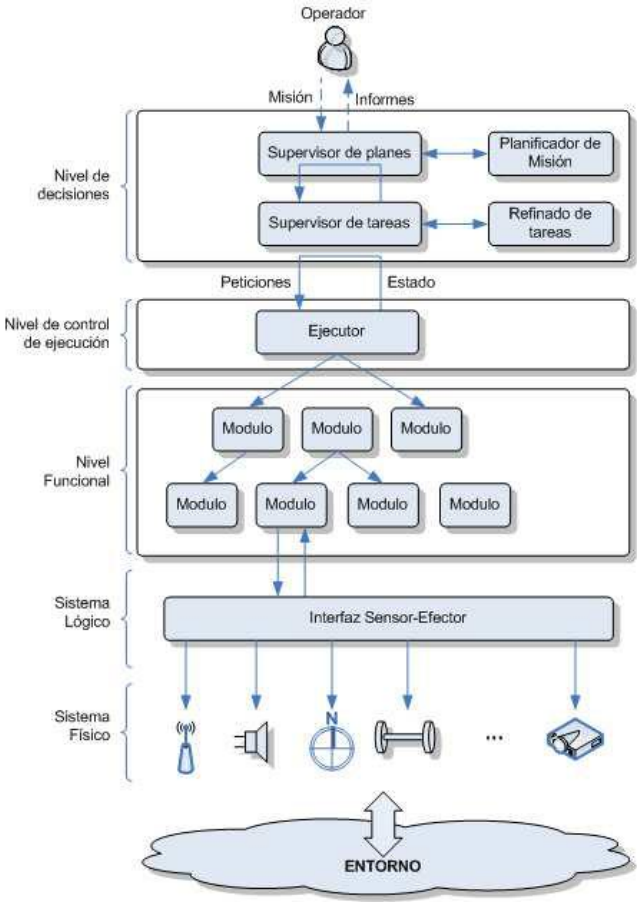



Figura 8. Arquitectura LAAS de control de la navegación de robots.

El nivel más bajo o cercano al hardware, y consiguientemente a la navegación reactiva, es el funcional; este nivel incluye las capacidades básicas del robot de acción y percepción. Estas capacidades están encapsuladas en unos módulos comunicados controlables. El siguiente nivel es el nivel de ejecución o ejecutivo. Este nivel, controla y coordina dinámicamente la ejecución de las funciones distribuidas en los módulos, de acuerdo con las tareas que se especifican en el nivel superior. El último de los niveles es nivel de decisión. En este nivel se incluyen las capacidades para producir las tareas del plan y supervisar su ejecución. Este nivel puede tener una gran cantidad de capas, de acuerdo con las capacidades que se le requieran. Realmente es un planificador y supervisor que permite integrar la reactividad y la deliberación.

## 2.3 PLANIFICACIÓN DE TRAYECTORIAS [MUÑOZ 95]

La planificación, ya sea global o local, consiste en encontrar una ruta segura capaz de llevar al vehículo desde la posición actual hasta la especificada de destino. El concepto de ruta segura implica el cálculo de un camino al menos continuo en posición, que sea libre de obstáculos. En virtud de esta ruta, el generador construirá las referencias que se le entregan al control de movimientos. Por ello, en la especificación de esta ruta se obvian las características cinemáticas y dinámicas del vehículo, ya que el cómputo de una referencia adecuada que cumpla con estos atributos es tarea del generador de caminos. Por tanto, la ruta al tan sólo asegurar continuidad en posición, supone que únicamente los robots móviles omnidireccionales puedan seguir una referencia de tales características.

### 2.3.1 Planificación por Grafos de Visibilidad

Los grafos de visibilidad **[LOPEZ 05]** proporcionan un enfoque geométrico útil para resolver el problema de la planificación. En esta técnica se supone un entorno bidimensional en el cual los obstáculos están modelados mediante polígonos. Para la generación del grafo este método introduce el concepto de *visibilidad*, según el cual define dos puntos del entorno como *visibles* si y solo si se pueden unir mediante un segmento rectilíneo que no intercepte ningún obstáculo (si dicho segmento resulta tangencial a algún obstáculo se consideran los puntos afectados como visibles). En otras palabras, el segmento definido debe yacer en el espacio libre del entorno *Cl*.
Así, si se consideran como nodos del grafo de visibilidad la posición inicial, la final y todos los vértices de los obstáculos del entorno, el grafo resulta de la unión mediante arcos de todos aquellos nodos que sean visibles.

En la Figura 9 se muestra el grafo de visibilidad construido merced a los obstáculos poligonales existentes en el entorno y las configuraciones inicial $q_a$ y final $q_f$.

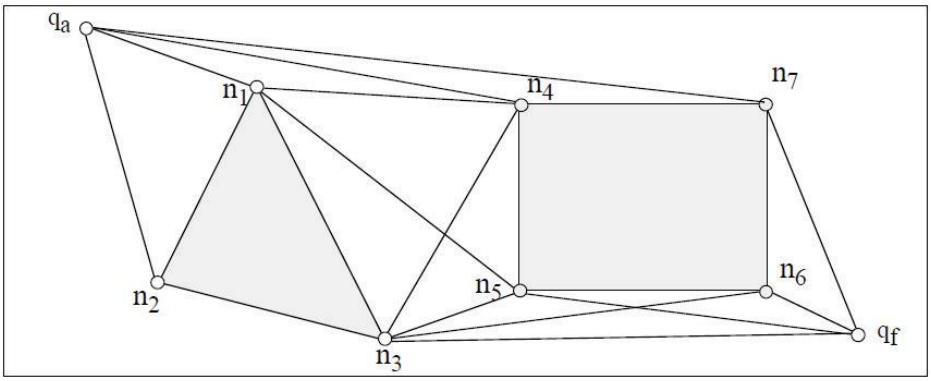


Figura 9. Grafo de visibilidad en un entorno de dos obstáculos

### 2.3.2 Planificación por Diagramas de Voronoi

La planificación por diagramas de Voronoi construye la ruta manteniendo una distancia frente a los obstáculos presentes en el entorno. Este método construye el lugar geométrico de la trayectoria manteniendo una igualdad de distancia entre los dos obstáculos más cercanos y el camino del robot. Estos diagramas son compuestos por segmentos rectos que se encuentran en una configuración geométrica donde las aristas de dos obstáculos son iguales, es decir presentan la misma distancia; y por segmentos parabólicos que es para el caso de un vértice y una arista.

En la Figura 10 se muestra un entorno delimitado por un polígono de aristas $\{e_1,e_2,e_3,e_4\}$ y un obstáculo triangular de vértices $\{n_1,n_2,n_3\}$ y aristas $\{a_1,a_2,a_3\}$. La retracción del espacio libre en una red continua de curvas es el diagrama de Voronoi $C_v$, representado mediante las líneas de trazo grueso. Los dos tipos de segmento utilizados en la construcción del diagrama pueden distinguirse en la mencionada figura, así, el segmento $s_1$ es el lugar geométrico de los puntos equidistantes entre la arista $e_1$, y el vértice $n_2$. Por otra parte, puede observarse como el segmento rectilíneo $s_2$ cumple la misma condición pero con respecto a las aristas $e_1$ y $e_2$.

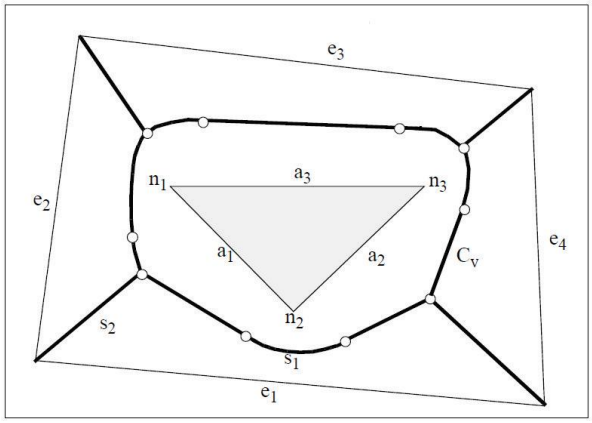


Figura 10. Retracción del espacio libre en un diagrama de Voronoi.

### 2.3.3 Planificación por Descomposición de Celdas

Este tipo de planificación toma un espacio libre y lo descompone en celdas para crear un camino conformado por una sucesión de celdas que no presentan discontinuidades, logrando crear una ruta de un punto inicial $q_a$ a uno final $q_f$, en este método se precisan dos problemas, el primero de ellos es la construcción de las celdas del espacio libre, que posean una forma geométrica con la que resulte fácil calcular un camino entre el punto inicial y el final, además determinar si las celdas del espacio libre son adyacentes. El segundo problema se centra en la construcción del grafo de conectividad de las celdas el cual es solucionado empleando un algoritmo que detecte las celdas que conectan el punto de partida y el de llegada.

En la Figura 11 se puede apreciar un ejemplo de Planificación de un trayectoria por descomposición de celdas.

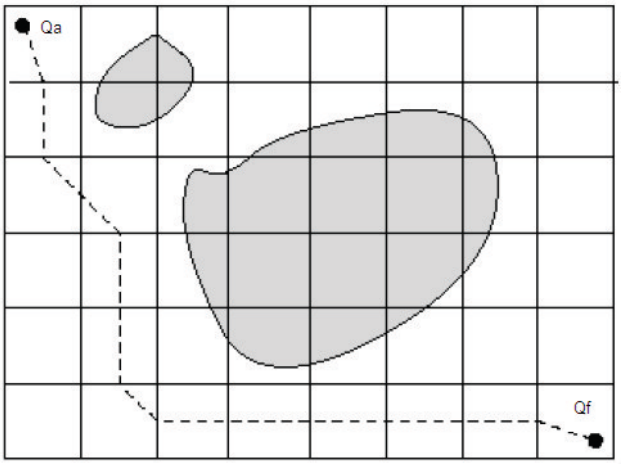


Figura 11. Planificación de una trayectoria por descomposición de celdas.

### 2.3.4 Planificación por Campos de Potencial

Los métodos basados en campos potenciales poseen una concepción totalmente distinta a los expuestos más arriba al estar basados en técnicas reactivas de navegación. El ámbito de uso de esta técnica se centra en la planificación local en entornos desconocidos, como puede ser el sorteo en tiempo real de obstáculos o de los que no se tiene constancia **[BORENSTEIN 89]** y **[KOREN 91]**.

La teoría de campos potenciales considera al robot como una partícula bajo la influencia de un campo potencial artificial, cuyas variaciones modelan el espacio libre. La función potencial $U$ en un punto *p* del espacio euclídeo, se define sobre el espacio libre y consiste en la composición de un potencial atractivo $U_a(p)$, que atrae al robot hacia la posición destino, y otro repulsivo $U_r(p)$ que lo hace alejarse de los obstáculos, es decir:

$$U(p) = U_a(p) + U_r(p)$$

La fuerza artificial $F(p)$ a la que afecta el vehículo en la posición *p*, por el potencial artificial $U(p)$ resulta:

$$F(p) = -\nabla U(p)$$

Al igual que la función potencial, la fuerza artificial es el resultado de la suma de una fuerza de atracción $F_a(p)$, proveniente de la posición destino, y otra fuerza de repulsión $F_r(p)$ debidas a los obstáculos del entorno de trabajo:

$$F(p) = F_a(p) + F_r(p)$$

Así, la navegación basada en campos potenciales se basa en llevar a cabo la siguiente secuencia de acciones:

i) Calcular el potencial $U(p)$ que actúa sobre el vehículo en la posición actual *p* según la información recolectada por los sensores.

ii) Determinar el vector fuerza artificial $F(p)$.

**iii)** En virtud del vector calculado construir las consignas adecuadas para los actuadores del vehículo que hagan que éste se mueva según el sentido, dirección y aceleración especificadas por $F(p)$.

## 2.4 REPRESENTACIÓN DEL ENTORNO

En la navegación de robots se distinguen fundamentalmente dos tipos de modelos: los mapas métricos y los mapas topológicos. Los mapas métricos estén basados en el sistema de referencia cartesiano del entorno y capturan las propiedades geométricas del mismo. Cada celda del mapa, que representa una posible localización del robot, se identifica mediante sus coordenadas en el entorno, y cubre un espacio generalmente pequeño cuyo tamaño define la granularidad del mapa. A mayor granularidad, mayor requerimiento de memoria para almacenarlo y mayor costo computacional para manejarlo. Los espacios relativamente grandes suponen mapas de dimensiones poco tratables. Si el robot está en una celda concreta del mapa, es relativamente sencillo seguir su trayectoria utilizando la odometría del robot e integrando sobre el desplazamiento, pero es bien conocido que estos sensores producen un error acumulativo no aceptable en largos recorridos **[BORENSTEIN 96] [EVERETT 95]**. Estos sensores sólo pueden medir el estado del robot en base al movimiento del propio robot, sin utilizar ninguna referencia externa y de ahí que en la práctica sea muy difícil corregir el error que se va acumulando por el propio deslizamiento de las ruedas, aceleraciones y deceleraciones del robot y la fricción. Para utilizarlos de manera fiable es necesario monitorizar y compensar continuamente el error de odometría **[THRUN 98]**.

Otra forma de modelar el entorno para la localización son los mapas topológicos, que consisten en una serie de nodos, cada uno correspondiente a un lugar del entorno, conectados mediante transiciones. Describen la conectividad entre las distintas localizaciones, representando el entorno en forma de una lista de lugares significativos conectados mediante arcos. La posición del robot relativa al modelo se determina en base a marcas o lecturas sensoriales diferentes. El principal problema es que la percepción de esas marcas sensoriales depende de la posición del robot y, por lo tanto, la localización puede fallar si el punto de vista del robot no es el adecuado. Su principal ventaja frente a los mapas métricos es la no dependencia de la posición geométrica y el bajo requerimiento de memoria y de cómputo. Entre las desventajas se debe mencionar que el “aliasing perceptivo" - la ambigüedad introducida por patrones sensoriales idénticos en posiciones diferentes - es mayor, debido a que el espacio que cubre cada nodo del mapa es mayor y, por lo tanto, resulta más difícil de caracterizar un lugar de forma única mediante los sensores.

Esta distinción de modelos es algo confusa ya que la gran mayoría de las aproximaciones topológicas utilizan información métrica **[SIMMONS 95]**. Es común combinar estos dos tipos de representaciones, de forma que se tenga una descripción métrica más fina de cada uno de los nodos de la representación topográfica **[DUCKETT 99] [THRUN 98]**.

Estas dos formas de representar el entorno predominaron en la literatura desde los años 80 hasta principios de los 90. Actualmente se han impuesto las rejillas de ocupación (occupancy grids) **[ELFES 89]**, también conocidas como modelos probabilísticos, que resultan de dividir el entorno en una rejilla discreta y asignar a cada celda un valor relacionado con la probabilidad de que cierta zona del espacio esté ocupada por un obstáculo. La probabilidad de ocupación se obtiene a partir del modelo sensorial. Por ejemplo, en el caso de un sensor de ultrasonido cuya lectura se traduce en la distancia aproximada a un objeto, es común aumentar la probabilidad de ocupación de las celdas próximas al objeto detectado y decrementar la ocupación de las celdas entre el robot y el objeto **[MARTIN 96]**. Cada celda representa un lugar ocupado, libre o desconocido según su probabilidad de ocupación. Es la técnica más utilizada debido a su simplicidad y su robustez. Desde su aparición estos modelos han evolucionado y existen múltiples variaciones en cuanto al modelo sensorial a utilizar para integrar la información que esos sensores proporcionan en el mapa. El movimiento del robot ofrece además información útil en el sentido en el que toda posición ocupada por el robot está libre inmediatamente después de que el robot realiza un movimiento. Al ser modelos geométricos, los requerimientos de memoria necesarios para almacenar el modelo pueden resultar excesivos dependiendo de la extensión del espacio y de la granularidad elegida. En **[DUCKETT 99]** se propone un método de aprendizaje basado en histogramas de rejillas de ocupación llamado OHM (Occupancy Histogram Matching); en lugar de almacenar toda la rejilla, se guardan únicamente los histogramas horizontales y verticales de la misma. De esta forma, se reducen las necesidades de cómputo para calcular las distancias a las marcas almacenadas. En **[DUCKETT 00]** se compara el OHM con diversos métodos de reconocimiento de marcas midiendo el nivel de incertidumbre en el entorno frente al costo computacional del proceso de reconocimiento de las marcas.

## 2.5 MARCAS DEL ENTORNO

Una marca característica es cualquier propiedad física que el robot puede percibir mediante sus sensores **[NEHMZOW 00]**. Una marca entonces no tiene por qué ser un objeto físico. Las marcas pueden ser propias del entorno (esquinas, líneas de

fluorescentes, tuberías, etc.) o artificiales, resultado de alterar el entorno de forma que el robot tenga a su alcance sensorial algún tipo de señal como códigos de barras, cartulinas de colores, etc. Las marcas también pueden ser patrones extraídos de imágenes visuales: brillo, características de objetos presentes en las imágenes como el color de las puertas, etc. Pueden ser también partes de las imágenes o incluso escenas **[TRAHANIAS 99]**. En los últimos años se ha desarrollado una tendencia al uso de sistemas de visión omnidireccionales como alternativa a las cámaras tradicionales con un rango de visión limitado **[LAMBRINOS 00]**, **[POPESCU 03]**, **[FRANZ 98]**.

Independientemente de la naturaleza de las marcas, éstas pueden clasificarse según distintos criterios. Por un lado las maracas pueden ser:

- **Marcas locales:** son características de un lugar en concreto, pero no necesitan de un sistema de referencia global para reconocerse. Pueden ser por ejemplo las lecturas de los sensores de ultrasonido en la esquina de una habitación, o también una lectura de la brújula alterada por un campo magnético local.

- **Marcas globales:** son aquellas relacionadas con un sistema de referencia global como puede ser el norte magnético de la tierra, la posición del sol a una hora determinada o la posición (x,y) del robot con respecto a su posición inicial, durante su trayectoria.

Por otro lado, las marcas pueden ser:

- **Dinámicas:** aquellas marcas que persisten cuando el robot está en movimiento. Por ejemplo, si consideramos los pasillos como marcas, el reconocimiento de los mismos es más robusto si depende de esas lecturas en el tiempo mientras el robot está en movimiento.

- **Estáticas:** al contrario que las dinámicas, el movimiento continuo del robot perjudica el reconocimiento de estas marcas, ya que no son persistentes frente al mismo. Por ejemplo, un patrón específico de los sensores de ultrasonido que localiza al robot en una posición y ángulo concretos de un mapa métrico.

Esta clasificación de marcas no es disjunta. ¿Cómo elegir el conjunto de marcas adecuado para una localización y navegación eficaz? es una pregunta abierta. La percepción del robot es muy distinta a la del humano y por ello, identificar el conjunto de marcas adecuado supone conocer de forma precisa el sistema sensorial del robot y el entorno y puede resultar más adecuado que sea el propio robot el que aprenda las marcas del entorno que le permitan una mejor localización **[GREINER 94], [THRUN 98]**.

El *aliasing* es un problema latente dependiendo del tipo de marcas utilizadas. Distintos objetivos pueden ser percibidos de la misma manera por el robot. Es posible combinar la información de marcas con la odometría para contrarrestar las ambigüedades entre localizaciones con el mismo patrón sensorial **[THRUN 98]; [BURGARD 98]**. En **[OWEN 98]** se presenta una aplicación para la construcción de mapas que permite al robot planificar rutas alternativas. En **[MORENO 03]** se utiliza Algoritmos Genéticos para corregir la posición y la orientación del robot a partir de las lecturas de los sónares. Una aproximación diferente es la propuesta en **[MORENO 02]** que permite al robot identificar las distintas oficinas en base al reconocimiento de los paneles que

contienen los nombres de los ocupantes. En **[RIZZI 01]** se presenta un método para la ecualización de la descripción de marcas utilizando únicamente información visual para el problema de robot homing, y en **[BENGTSSON 03]** se aproxima de manera teórica la localización del robot basándose en el emparejamiento de lecturas de ultrasonidos.

## 2.6 NAVEGACIÓN TOPOLÓGICA

Actualmente la navegación topológica está considerada como la forma más simple y natural de navegación autónoma en robots móviles. La navegación topológica utiliza una representación topológica del entorno en el que se trabaja.

Un ejemplo de este tipo de navegación es la capacidad de enviar a un robot un comando del tipo “sigue el pasillo y al final gira hacia la izquierda y entra en la tercera habitación de la derecha”. Incluso sin un mapa, el robot tiene suficiente información como para guiarse, a través de los pasillos indicados sin perderse, siguiendo las marcas que pueden ser “pasillo”, “final”, “tercera habitación”.

A continuación se citan los diferentes tipos de navegación topológica:

- **Relacional**
  La navegación topológica relacional es la técnica más utilizada y se basa en una serie de mapas en los cuales los puntos clave son los nodos de un grafo relacional; y el camino para llegar de un nodo a otro son las aristas de dicho grafo (Figura 12).

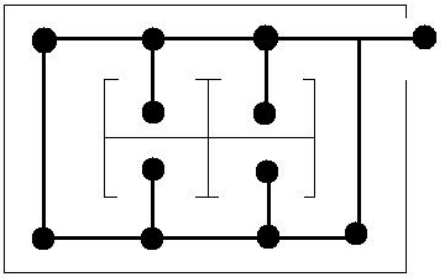


Figura 12. Representación relacional del plano de una planta

- **Asociativa**
  Al contrario que la técnica relacional, las técnicas asociativas se basan fundamentalmente en crear un comportamiento que traduce un estímulo observado en una dirección de movimiento para alcanzar el patrón o marca observado.

  Debido a que las técnicas relacionales utilizan una representación explícita, pueden apoyar la planificación de ruta. Las técnicas asociativas son mejores para retraer los caminos conocidos.

### 2.6.1 Puntos de referencia y Puertas de enlace

La Navegación Topológica depende de la presencia de marcas (*landmarks*). Una marca es uno o más rasgos característicos de un objeto o lugar de interés. Hay que tener en cuenta que no es necesariamente un objeto único, autónomo como "puerta

roja". Una marca o punto de referencia puede ser tanto una agrupación de objetos, por ejemplo "McDonald's", un edificio alto y luminoso de una forma determinada, y un estacionamiento con mucha actividad.

Las marcas se representan como nodos de un grafo (que pueden tener características asociadas, para garantizar su identificación unívoca), los cuales se unen por los arcos que normalmente representan la accesibilidad (si existe arco entre dos nodos, el robot puede desplazarse directamente de uno a otro de los *landmarks* a los que los nodos representan). Estos arcos pueden también estar etiquetados con características del recorrido como distancia, dirección, tiempo de tránsito, etc.

Las marcas son utilizadas en casi todos los aspectos de la navegación. Si un robot encuentra una de estas en el entorno y esa aparece en el mapa, entonces el robot se localiza en relación con el mapa. Si el robot planea una ruta formada por segmentos, los puntos de referencia son necesarios para que el robot pueda decir cuando se ha completado un segmento y cuando otro debe empezar. Si un robot encuentra nuevos puntos de referencia, estos se pueden añadir a su memoria espacial, creando y ampliando un mapa.

En **[KORTENKAMP 94]** se publicó un caso especial de marcas denominadas: *gateways* (Puertas de enlace). Un *gateway* es una oportunidad para que un robot cambie su dirección general de la navegación. Por ejemplo, una puerta es el cruce de dos pasillos y el robot puede optar por seguir derecho o girar o pasar a través de ellos. Debido a que los *Gateway* son oportunidades de navegación, el reconocimiento de estos es fundamental para la localización, la planificación de ruta y la cartografía.
Las marcas pueden ser artificiales o naturales. Los términos "artificial" y "natural" no deben confundirse con "lo hecho por el hombre" y "lo orgánico". Una marca artificial es un conjunto de características añadidas a un objeto existente o localidad con el fin de apoyar el reconocimiento tanto de la marca o alguna otra actividad perceptiva. La señal de salida de la autopista interestatal es un ejemplo de un punto de referencia artificial. Se puso en su lugar con el propósito de ser fácil de ver (retro-reflexivo) y el tipo de letra en blanco sobre fondo verde está dimensionada para una visibilidad óptima (la actividad perceptiva es la lectura de la señal). Un punto de referencia natural es una configuración de características existentes seleccionadas para el reconocimiento el cual no fue expresamente designado para la actividad perceptual. Si alguien da instrucciones a su casa, "tomar la segunda cuadra a la derecha después de McDonald," McDonald está siendo utilizado como una señal de orientación para la navegación a su casa. Es evidente que McDonald's no se construyó con el propósito de ser una señal de navegación a una casa particular. El hecho que fue utilizado para otro propósito significa que es un punto de referencia natural.

Independientemente de si una marca es artificial o natural, debe cumplir tres criterios:

- *Ser fácilmente identificables.* Si el robot no puede encontrar el punto de referencia no es útil.

- *Apoyar la tarea dependiendo de la actividad.* Si la actividad de la tarea dependiente es simplemente una señal de orientación ("tome la segunda cuadra a la derecha después de McDonald's"), luego de ser reconocido es suficiente. Supongamos que un punto de referencia está destinado a proporcionar condiciones de orientar la estimación de acoplamiento del transbordador espacial a una estación espacial. En ese caso, la marca debería hacerse fácil para extraer la distancia relativa al contacto.

- *Ser perceptible desde muchos puntos de vista diferentes*. Si la marca no es ampliamente visible, el robot no puede encontrarla.

La Figura 13 muestra puntos de referencia artificiales construidos para ser usados en el Concurso de Robots Móviles AAAI en 1992 **[DEAN 93].** En esta prueba cada robot tenía que seguir una ruta entre cualquier secuencia de puntos de interés en un entorno. A los equipos se les permitió marcar los puntos de ruta con puntos de referencia artificiales. Cada punto de control es fácilmente reconocible por el patrón de tablero de ajedrez. La tarea depende de la actividad de ir al punto de referencia correcto que es facilitado por el código de barras cilíndricas que son únicos para cada estación. Hay que tener en cuenta que el uso de un cilindro garantiza que los puntos de referencia sean perceptibles desde cualquier punto de vista en el entorno.

Figura 13. Marcas Artificiales usadas en el concurso de Robots Móviles AAAI en 1992 (Fotografía cortesía de AAAI).

Una buena marca tiene muchas otras características deseables. Debe ser pasiva con el fin de estar disponible a pesar de una falla de energía. Esta debe ser perceptible en toda la gama donde el robot tal vez necesite ver. Debe tener características distintivas, y si es posible, con características únicas. Las características distintivas son las que son únicas a nivel local, estas aparecen sólo como parte de la marca de cada punto de vista del robot en esa región del entorno (por ejemplo, sólo hay un McDonald en Busch Boulevard). Si la función se produce sólo una vez en toda la región de operaciones (por ejemplo, sólo hay un McDonald's en Tampa), entonces esta característica sería considerada única en el mundo. Además de ser perceptible para los fines de reconocimiento, debe ser perceptible para la tarea. Si el robot debe ubicarse dentro de 0,5 metros del punto de referencia, entonces debe ser el punto de referencia destinado a lograr esa precisión.

### 2.6.2 Métodos De Navegación Topológica

#### *2.6.2.1 Métodos Relacionales*

Los métodos relacionales representan el entorno como un gráfico o una red de nodos y aristas. Los nodos representan las marcas o puntos de referencia. Las aristas representan una vía navegable entre dos nodos, en efecto, esos dos nodos tienen una relación espacial. La información adicional se puede adjuntar a las aristas, como la dirección (N, S, E, W), la distancia aproximada, el tipo de terreno, o los comportamientos necesarios para navegar por ese camino. Se puede calcular rutas entre dos puntos utilizando algoritmos de grafo estándar, tal como el algoritmo de Dijkstra.

Una de las primeras investigaciones de grafos relacionales para la navegación fue hecha por Smith y Cheeseman **[SMITH 86].** Ellos representan el entorno como un grafo relacional, donde las aristas representan la dirección y la distancia entre nodos. En la Figura 14 se muestra un ejemplo de este tipo de grafo.

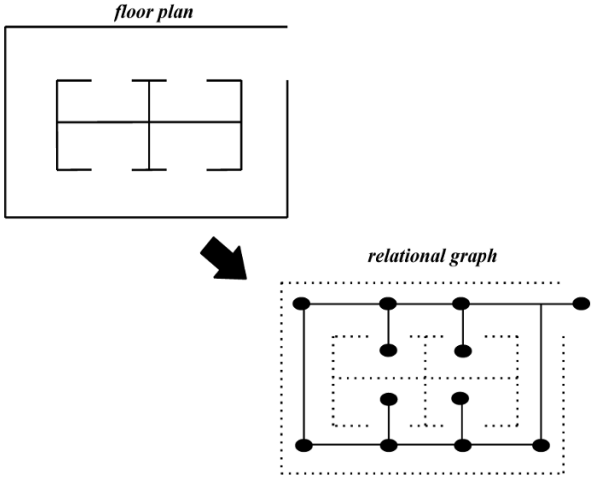



Figura 14. Representación de un plano de planta como un grafo relacional.

#### *2.6.2.1.1 Lugares Distintivos*

Kuipers y Byun vincularon grafos relacionales a la detección en sus trabajos iniciales con lugares distintivos **[KUIPERS 91].** Un lugar distintivo es una marca que el robot puede detectar desde una región cercana llamada sector. Su trabajo fue motivado por la investigación en ciencia cognitiva indicando esa representación espacial en el reino animal formando una jerarquía de múltiples niveles. (En estudios más recientes, esta jerarquía no es tan claramente dividida como se pensaba anteriormente.) En la Figura 15 se muestra un ejemplo de esta jerarquía, donde el nivel más bajo, o la forma más primitiva de representar el entorno, es mediante la identificación de puntos de referencia (puertas, pasillos) y el conocimiento procedimental para viajar entre ellos (seguimiento de salón, moverse a través de las puertas). La siguiente capa es topológica. Esta representa a los puntos de referencia y el conocimiento de procedimiento en un grafo relacional, que apoya la planificación y el razonamiento. El nivel superior es métrico, donde el agente había aprendido las distancias y orientaciones entre los puntos de referencia y podría colocarlos en un sistema de coordenadas fijas. Las capas superiores representan un aumento de la inteligencia.

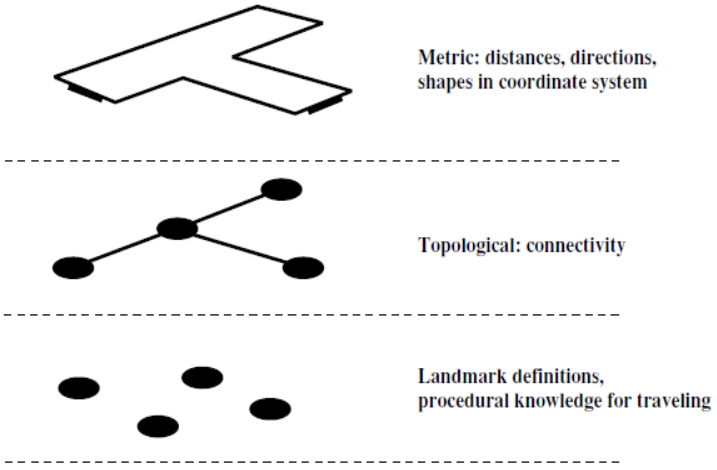



Figura 15. Jerarquía espacial de múltiples niveles, después de Byun y Kuipers. **[KUIPERS 91]**

La representación de Kuipers y Byun es de un interés particular. Cada nodo representa un lugar diferente. Una vez en el sector, el robot puede posicionarse en un lugar conocido en relación a la marca mediante la lectura de los sensores. Un ejemplo de un lugar distintivo es un rincón (Kuipers y Byun trabajaron en la simulación; Esto no resultó ser realista con sonares). La idea era que el robot pudiera moverse repetiblemente en el sector de la esquina hasta, por ejemplo, 1 metro de cada pared. Luego el robot hubiera sido localizado en el mapa.

Un arco o borde en el grafo relacional se llama una estrategia de control local, o ***lcs***. La estrategia de control local es el procedimiento para llegar del nodo actual al siguiente nodo. Cuando los robots detectan la señal que se está llenando en los valores de un conjunto de características. El robot utiliza un algoritmo de escalada (***hill-climbing algorithm***). El algoritmo de escalada dirige al robot alrededor de todo el sector hasta que una función de medición (por ejemplo, a qué distancia de las paredes está) indica cuando el robot está en una posición en la que se maximizan los valores de la función (por ejemplo, ambos son de 1 metro de distancia).

El punto donde los valores de característica se aprovechen al máximo es el lugar distintivo. El algoritmo ***hill-climbing*** es un enfoque muy simple. La idea es que para la mayoría de colinas, si se elige siempre el paso siguiente que es la más alta (no se puede mirar hacia adelante), entonces se llega a la cima rápidamente. Por lo tanto, el robot se mueve siempre en la dirección que aparece al hacer el cambio más positivo en la función de medición.

Considere la posibilidad de un robot de navegar por un pasillo ancho en un callejón sin salida como se muestra en la Figura 16. La estrategia de control local es un comportamiento, tal como seguimiento de salón, que opera en conjunto con un liberador, mirar para la esquina. El disparador es una señal exteroceptiva. Cuando se activa, el robot se encuentra en el sector del lugar distintivo, y el comportamiento liberado, subir la pendiente hacia la esquina (distancia = 1), dirige el robot a 1 metro de cada pared.

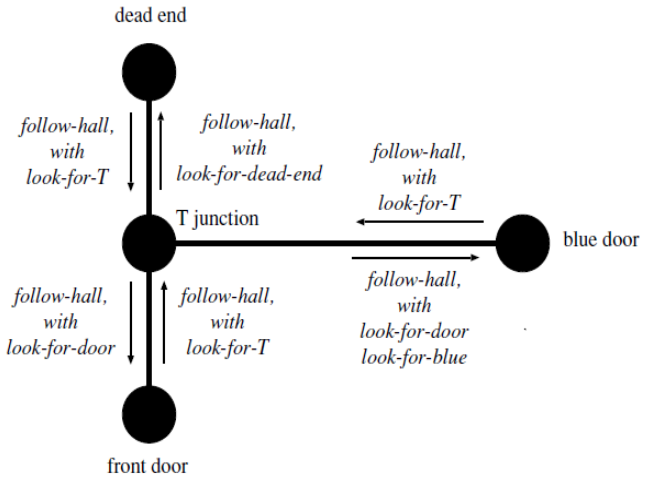


Figura 16. Comportamientos que actúan como estrategias de control local, y liberadores como medios de señalización a la entrada de un barrio.

### *2.6.2.2 Métodos Asociativos*

Los métodos asociativos para la navegación topológica esencialmente crean un comportamiento que convierte las observaciones del sensor en la dirección para llegar a un punto de referencia particular. La suposición subyacente es que una marca o ubicación de interés para la navegación normalmente tiene dos atributos:

- *Estabilidad Perceptiva*. Que los criterios de la ubicación que están muy juntos deben ser similares.

- *Distinguibilidad perceptual.* Que los criterios de distancia deben ser distintos.

#### *2.6.2.2.1 Visual Homing*

Esta técnica está basada en el estudio de Randal C. Nelson [**RANDAL 97**] sobre la navegación de las abejas. Consiste en crear firmas de las imágenes tomadas, es decir, dividir una imagen en secciones, las cuales serán examinadas para obtener una determinada medida (densidad de bordes en la sección, orientación dominante de los bordes, intensidad media de los píxeles, etc.). La firma de una imagen constituye un patrón (ver ejemplo en la Figura 17).

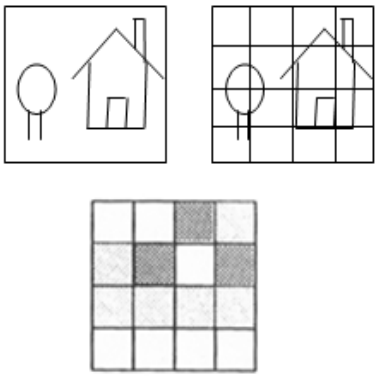


Figura 17. Obtención de la firma de una imagen (patrón).

Si un robot es capaz de identificar la firma de una imagen, o un trozo de ella, en la imagen actual, podrá entonces girar para localizarse en relación al lugar. Este uso de patrones para dirigir a un robot hacia un lugar específico se denomina *visual homing*. En la Figura 18 se muestra la firma obtenida a partir de dos imágenes y el comportamiento asociado a cada una de ellas: la primera le indica al robot que debe seguir de frente y la segunda le indica que debe girar hacia la derecha para alinearse frente al objetivo al que debe llegar.

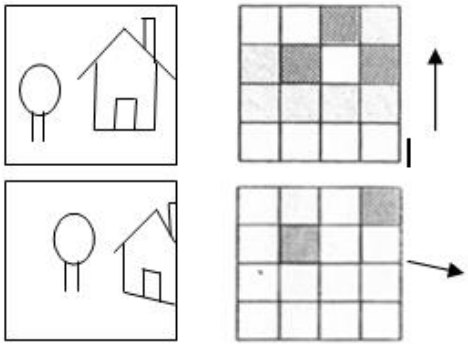


Figura 18. Firma de dos imágenes con sus respuestas asociadas

#### *2.6.2.2.2 QualNav*

Levitt y Lawton **[LEVITT 90]** tomaron las ideas de los sectores y Visual Homing a un extremo para la navegación al aire libre a grandes distancias, como parte de la Agencia de Proyectos de Investigación Avanzada de la Defensa (DARPA), en Vehículos Terrestres Autónomos (ALV) a finales de 1980 **[LEVITT 90]**. En ese momento, la navegación topológica utilizando grafos relacionales parecía prometedora para los ambientes interiores, pero parecía resistirse a aplicaciones en terrenos al aire libre. Parte del desafío fue la noción de localización. Con ese desafío surgió la pregunta ¿Qué podría el robot usar como una marca?. Era de esperarse que el vehículo terrestre fuese capaz de viajar a 50 millas deliberadamente evitando lugares obvios tales como casas, cruces de carreteras y torres de radio. La idea era tener un vehículo robot capaz de sigilo de exploración de un área y ver si las fuerzas enemigas estaban cerca. El vehículo robot tiene que dar alguna indicación de dónde se encuentran las fuerzas enemigas, así como el regreso a la base.

Dentro de tratar de localizar el vehículo a una marca particular, la idea detrás de QualNav **[LEVITT 90]** (un acrónimo de "navegación cualitativa") fue para localizar el vehículo a una particular región de orientación, o una parte del entorno. La región de orientación está definida por límites pares de marcas. Un límite de una marca es una línea imaginaria trazada entre dos puntos de referencia. Como se observa en la Figura 19, el trazado de líneas entre todos los pares de marcas candidatas particionarán el entorno en regiones de orientación, a pesar de que las regiones de orientación puedan ser de forma muy extraña.

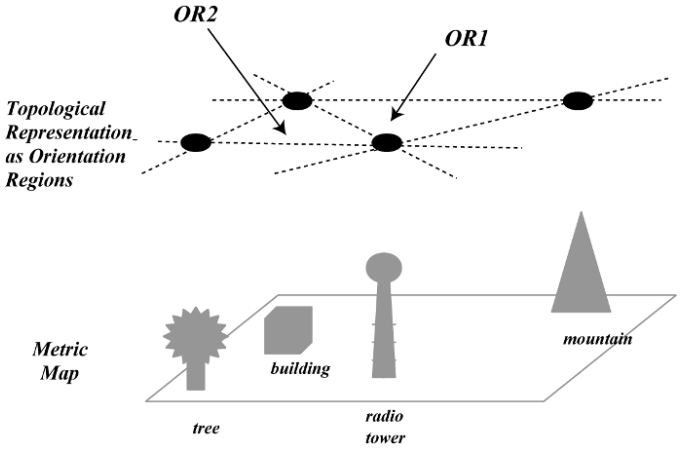


Figura 19. División de un espacio al aire libre en las regiones de orientación, después de Levitt y Lawton. **[LEVITT 90]**

Una región de orientación es conceptualmente similar a un sector. Dentro de una región de orientación, todos los puntos de referencia aparecen en la misma relación. Si el vehículo está en OR1, la marca se encuentran en el mismo orden si el robot gira en la misma dirección (ya sea siempre hacia la derecha o siempre hacia la izquierda): *building-mountain-tower*. Si el vehículo está en OR2, los puntos de referencia serán *tree-building-intersection*.

Una cualidad interesante de las regiones de orientación y los límites par de referencia es que el vehículo directamente puede percibir cuando ha entrado en una nueva región de orientación. Por ejemplo, como el robot se mueve de OR1 a OR2, el límite de par de marca *building-tower* se encuentra en frente de él, entonces está en ambos lados del vehículo, y finalmente detrás de él. La transición es un caso de percepción, y no requiere de conocimientos sobre la distancia de los puntos de referencia, al igual que la relación de los puntos de referencia para el robot solo haya cambiado.

El uso de las regiones de orientación permite al robot crear un mapa topológico en exteriores, para que explore el entorno, o para localizarse en un mapa métrico. El robot no tiene que ser capaz de estimar la distancia a cualquiera de los puntos de referencia.

Como el robot se desplaza dentro de una región de orientación tiene la característica de *Visual Homing.* El robot tal vez necesite volver sobre su camino con la mayor precisión posible a través de una región de orientación. (Un robot militar que tenga que regresar sin ser visto). Sin saber el rango de las marcas que delimitan la región de orientación, el robot no puede hacer nada. Pero si tiene que recordar los ángulos de cada marca cada n minutos, se puede mover para seguir los ángulos. El conjunto de ángulos recordados en un punto a lo largo de la ruta se llama *Viewframe*.

Los métodos asociativos son interesantes debido a la estrecha conexión entre la detección de buscador de blancos. Los conceptos de firma de una imagen y *Viewframe* no requieren que el robot reconozca explícitamente lo que es un punto de referencia, sólo que es estable y perceptivamente distinguible de la región de interés. Desafortunadamente los métodos asociativos requieren el almacenamiento masivo y son frágiles en la presencia de un entorno dinámico en el que los puntos de referencia

pueden ser ocluidos o cambian de repente. Por supuesto, esto es más problemático para los ambientes interiores que los exteriores.

La navegación orientada a un objetivo o propósito es sin duda uno de los problemas principales a resolver para el desarrollo de la autonomía de robots móviles. Supone un paso más allá de la mera supervivencia. La navegación es por lo tanto una competencia básica de la que todo robot móvil debe estar dotado. Para que el robot pueda navegar, debe mantener información sobre la posición o estado actuales, además de información sobre el objetivo y la relación entre el estado actual y ese objetivo.

Tradicionalmente, tanto en los sistemas deliberativos como en sus sucesores híbridos, la navegación se define como la combinación de tres tareas fundamentales: La auto-localización, la planificación de trayectorias y la construcción e interpretación de mapas. La navegación se fundamenta mediante un modelo centralizado del mundo que, o bien se le proporciona, o bien aprende el robot. Ese modelo centralizado se denomina mapa, aunque difiere del concepto de mapa que los humanos utilizamos.

Los mapas, métricos o topológicos, han evolucionado hacia modelos probabilísticos, que constituyen hoy en día la técnica dominante a la hora de representar el entorno en este tipo de sistemas.

## 2.7 VISIÓN ARTIFICIAL

La visión en robótica es el campo de la informática que estudia el uso de cámaras como sensores. Su función principal es reconocer y localizar objetos en el entorno mediante el procesamiento de las imágenes. La visión artificial estudia estos procesos para construir máquinas con capacidad de “entender” una escena o las características de una imagen. Con la incorporación de cámaras en los robots se obtiene un sensor capaz de procesar una muy alta cantidad de datos a bajo coste económico. Sin embargo, la extracción de la información a partir de dichos datos es compleja, lo que implica un alto coste computacional.

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes.
- Identificar en diferentes imágenes una misma escena u objeto.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la escena; tal modelo podría ser usado por un robot para navegar por dicha escena.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones, aprendizaje estadístico, geometría de proyección, procesado de imágenes, teoría de grafos y otros campos.

### 2.7.1 Procesamiento Digital De Imágenes

El procesamiento digital de imágenes desarrolla las bases teóricas y algorítmicas por medio de la cuales se extrae información del mundo real, de manera automática a

partir de la captura de una imagen. Esta información puede relacionarse con el reconocimiento de objetos genéricos, descripciones tridimensionales del mismo, posición y orientación del objeto.

A continuación se describen los pasos para realizar el procesamiento digital de imágenes (Figura 20).

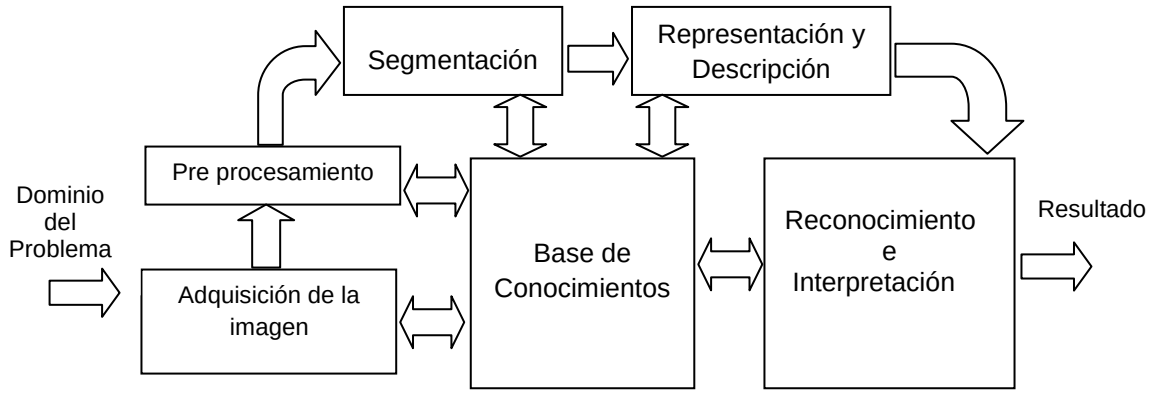



Figura 20. Pasos fundamentales del procesamiento de imágenes

### *2.7.1.1 Adquisición de la Imagen*

Las imágenes digitales se adquieren por medio de un dispositivo físico que sea sensible a la banda visible del espectro electromagnético y que produzca una señal eléctrica proporcional al nivel de energía percibido. Luego esta señal es transmitida a un digitalizador, el cual se encarga de convertir la señal eléctrica a una forma digital aunque en la actualidad existen cámaras que producen una señal digital (Estándar IEEE 1394) que puede ser transmitida a la computadora sin necesidad de un digitalizador. Posteriormente por medio de un software (driver) se enlazan los programas de aplicación con el hardware de captura. Una vez enlazados, este software permite manipular las características de la cámara como ajustar el brillo, el contraste, etc.

La transformación de una imagen analógica a otra discreta se llama digitalización y es el primer paso en cualquier aplicación de procesamiento de imágenes digitales.

### *2.7.1.2 Pre procesamiento*

Después de que la imagen digital ha sido obtenida, el siguiente paso es el pre-procesamiento. El pre-procesamiento típicamente trata con técnicas para realzar el contraste, remover ruido y es muy útil porque ayuda a suprimir información que no es relevante para los objetivos particulares de análisis en un caso dado. Por lo tanto, el objetivo del pre-procesamiento es una mejora de los datos de la imagen que suprima las distorsiones indeseadas e incremente las características relevantes para su posterior procesamiento. Algunas técnicas de pre-procesamiento son:

***Realce de imagen.*** Las técnicas de realce pretenden aumentar el contraste entre los objetos presentes en la imagen, esta técnica no aumenta la calidad radiométrica, sino que mejora algunas de sus características visuales para las siguientes etapas del análisis automático de las imágenes. **[PLATERO 05]**

***Restauración de imágenes.*** La restauración de imágenes consiste en recuperar, de la mejor manera posible la imagen original, a partir de su correspondiente imagen

degradada y del conocimiento a priori de las causas de la degradación. El conocimiento de la degradación puede ser obtenido ya sea a partir de un modelo analítico a bien a partir de un modelo empírico. **[FERGUS 05]**

***Modelos de representación.*** Para la representación de una imagen real en formato digital, se utilizan diversos modelos para facilitar la especificación de los colores de una forma normalizada. Algunos ejemplos de Modelos son: RGB (usado en monitores, cámaras), CMYK (usado en impresoras), HSI (usado para procesar imágenes). **[LOAIZA 11]**

***Conversión RGB a HSI.*** Uno de los modelos más utilizados para visualizar imágenes es el modelo RGB, aunque no es el idóneo para el análisis basado en color de imágenes, debido a que los factores como la iluminación afectan fuertemente este modelo, por eso es muy común encontrar sistemas que realicen la conversión del modelo RGB a HSI ya que este último es ideal para trabajar en ambientes de mala iluminación, esta utilidad se debe a que: El componente de intensidad de iluminación (I), se desacopla de la información de color, el componente matiz (H) se relaciona con el color y la saturación (S) con la cantidad de luz blanca.

***Transformación de intensidad.*** En una transformación de intensidad se modifica el valor del píxel sin importar su posición dentro de la imagen, algunas transformaciones de intensidad incluyen modificar el brillo, contraste, matiz, saturación, gamma, etc.

***Las transformaciones geométricas.*** Son comunes en gráficas por computadora y también son utilizadas en análisis de imágenes. Permiten la eliminación de distorsiones geométricas que ocurren cuando una imagen es capturada, como por ejemplo las provocadas por los lentes de la cámara. Una transformación geométrica es una función T que coloca al píxel (x,y) en una nueva posición (x ',y').
En la Figura 21 se puede apreciar un ejemplo de Transformación geométrica en un plano.

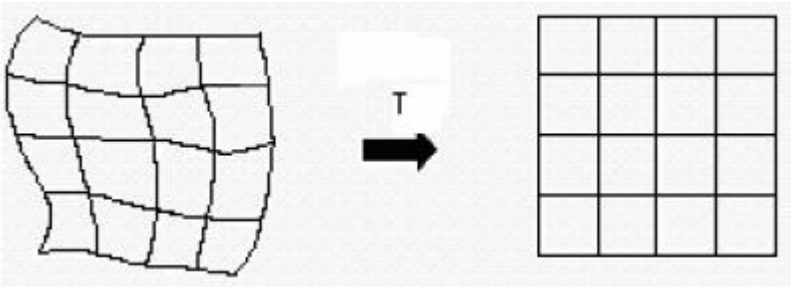


Figura 21. Transformación geométrica en un plano

### *2.7.1.3 Segmentación*

La segmentación particiona una imagen de entrada en sus partes constituyentes con el fin de obtener los objetos de interés de la escena. Generalmente, la segmentación automática es una de las tareas más difíciles en el procesamiento de imágenes.

Los algoritmos de segmentación se basan en una de las dos propiedades básicas de los valores del nivel de gris: discontinuidad o similitud.

***Discontinuidad***. Se divide la imagen basándose en cambios abruptos de nivel de gris: *Detección de puntos aislados:* Una máscara para detectar un punto aislado es la siguiente:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

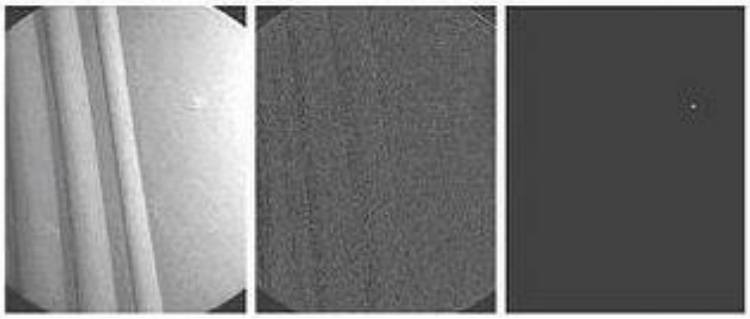


Figura 22. Ejemplo de una máscara para detectar un punto aislado.

*Detección de líneas:* Para la detección de líneas de un píxel de ancho, podemos utilizar una máscara 3×3 que se irá moviendo por los píxeles de la imagen. Los píxeles que forman parte de una línea horizontal, vertical o diagonal, tendrán respuestas extremas ante alguna de las máscaras mostradas en la Figura 23.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | -1 | 2 | -1 | -1 | -1 | 2 | -1 | -1 | -1 | 2 |
| 2 | 2 | 2 | -1 | 2 | -1 | -1 | 2 | -1 | -1 | 2 | -1 |
| -1 | -1 | -1 | -1 | -1 | 2 | -1 | 2 | -1 | 2 | -1 | -1 |

Figura 23. Máscaras 3x3 para la detección de líneas.

*Detección de bordes:* Se centra en la detección de contornos; delimitan el borde de un objeto y segmentan los píxeles dentro del contorno como pertenecientes a ese objeto. Su desventaja consiste en conectar contornos separados o incompletos, lo que los hace susceptibles a fallas.

***Similitud***. Se divide la imagen basándose en la búsqueda de zonas que tengan valores similares, conforme a unos criterios prefijados:
*Crecimiento de región:* Es un procedimiento que agrupa los píxeles o subregiones de la imagen en regiones mayores basándose en un criterio prefijado.
*Umbralización:* Un método básico para diferenciar entre uno o varios objetos y el fondo de la imagen es a través del histograma, con el cual se obtiene una gráfica donde se muestran el número de píxeles por cada nivel de gris que aparece en la imagen. Luego para binarizar la imagen, se deberá elegir un valor adecuado (umbral) dentro de los niveles de grises, de tal forma que el histograma forme un valle en ese nivel. Todos los niveles de grises menores al umbral calculado se convertirán en negro y todos los mayores en blanco.

#### *2.7.1.4 Representación y Descripción*

Es llamada también selección de características y trata de la extracción de los rasgos que resulta en alguna información cuantitativa de interés o características que son básicas para diferenciar una clase de objetos con otra.

Algunos esquemas de representación son: códigos de cadena, aproximaciones poligonales y esqueleto de una región.
Los *códigos de cadena* son usados para representar una frontera mediante una secuencia de segmentos conectados de línea recta.

*Aproximación poligonal* es capturar la esencia de la figura con el mínimo posible de segmentos poligonales. Se basa en el principio de que un borde de píxeles consecutivos puede ser aproximado con cierto grado de precisión por un polígono.

*Esqueleto de región* es la representación de la forma estructural de la figura reducida a su gráfica. Una figura en dos dimensiones puede ser representada por sus ejes medios (esqueleto) y las distancias de los ejes a su borde.

#### *2.7.1.5 Reconocimiento e Interpretación*
El reconocimiento es el proceso que etiqueta, o asigna un nombre, a un objeto basándose en la información que proveen sus descriptores. La interpretación involucra la asignación de significado a un conjunto de objetos reconocido. El reconocimiento de patrones comprende técnicas para asignar cada patrón a su respectiva clase de manera automática sin intervención humana.

Existen tres principales tipos de patrones, los vectores (para descripciones cuantitativas), las secuencias y los árboles (para descripciones estructurales):

- Los *patrones de vector* son representados por letras minúsculas en negrillas y toman la forma:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- Las *secuencias de descripción* son útiles en objetos o entidades cuya estructura está basada en una conexión de primitivas simples generalmente asociadas a su forma. La secuencia se forma ordenando las primitivas de manera tal que describan al patrón.

- Los *árboles*, se utilizan para representaciones con una estructura más jerárquica. La raíz del árbol representa toda la escena, los niveles inferiores van clasificando la imagen en sus partes constituyentes hasta que se llega a un nivel en donde no se puede reconocer más.

### 2.7.2 Matching
El término *matching* se utiliza en visión artificial para establecer la correspondencia entre imágenes distintas de un mismo objeto. Comúnmente, se emplea como 'objeto' un punto, una línea o una región.

#### *2.7.2.1 Correspondencia entre puntos*
Una posibilidad la da el *flujo óptico*, que corresponde a un conjunto de vectores que indica el movimiento de una imagen hacia otra. En el flujo óptico primero se divide la primera imagen en ventanas pequeñas. Luego, para cada una de estas ventanas se

busca en la segunda imagen la ventana (de las mismas dimensiones) que sea lo más parecida posible a la ventana de la primera imagen. Finalmente, teniendo la ubicación de la ventana en la primera y en la segunda imagen, se obtiene un vector de desplazamiento. Este vector se puede interpretar como el movimiento que sufrió la ventana de la primera imagen para transformarse en la ventana de la segunda imagen.

Como se puede apreciar, este método sólo se puede usar si la diferencia entre las dos imágenes es pequeña, ya que para un movimiento mayor, por ejemplo una rotación de 90º, no será posible encontrar las ventanas similares.

Matemáticamente se puede establecer que si las imágenes son $I_1$ e $I_2$, cuyos tamaños son *N x M*, se puede dividir la imagen en ventanas de *n x m*. Siendo los índices $I$ = 1,…, *n* y *J* = 1,…*m* los que indican cuál de las ventanas de $I_1$ se va a analizar, se puede encontrar el vector de desplazamiento $v_{IJ}$ = ($k_I$, $k_J$) minimizando la función objetivo que calcula el valor absoluto de la diferencia entre la ventana (*I, J*) de la imagen $I_1$ con una ventana en la imagen $I_2$ desplazada $k_I$ píxeles y $k_J$ píxeles en la dirección *i* y *j* respectivamente, es decir:

$$k_I, k_J = \arg mín \sum_{i=1}^{n}\sum_{j=1}^{m}\left|I_1(nI+i, mJ+j) - I_2(nI+i+k_I, mJ+j+k_J\right|$$

En la Figura 24 se muestra a manera de ejemplo el vector de desplazamiento $v_{55}$. El vector de desplazamiento también se puede encontrar maximizando una función objetivo que calcule la correlación entre las dos ventanas:

$$k_I, k_J = \arg max \sum_{i=1}^{n}\sum_{j=1}^{m}\left|I_1(nI+i, mJ+j) I_2(nI+i+k_I, mJ+j+k_J\right|$$

o bien utilizando la correlación normalizada:

$$k_I, k_J = \arg max \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left|I_1(nI+i, mJ+j) I_2(nI+i+k_I, mJ+j+k_J\right|}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} I_1^2(nI+i, mJ+j)\sum_{i=1}^{n}\sum_{j=1}^{m} I_1^2(nI+i+k_I, mJ+j+k_J)}}$$

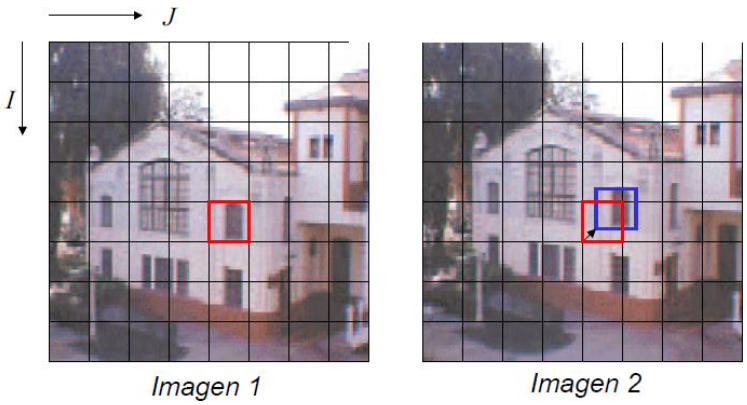



Figura 24. Flujo óptico: en la imagen 2 se busca la ventana roja más parecida a la ventana *I* = 5*; J* = 5 de la imagen 1.

En la Figura 25 se muestran dos imágenes en una esquina en las que se observa que ha habido un movimiento de los automóviles. Al calcular el flujo óptico, sólo en las

porciones en que ha ocurrido un cambio se detectará un vector de desplazamiento mayor que cero.

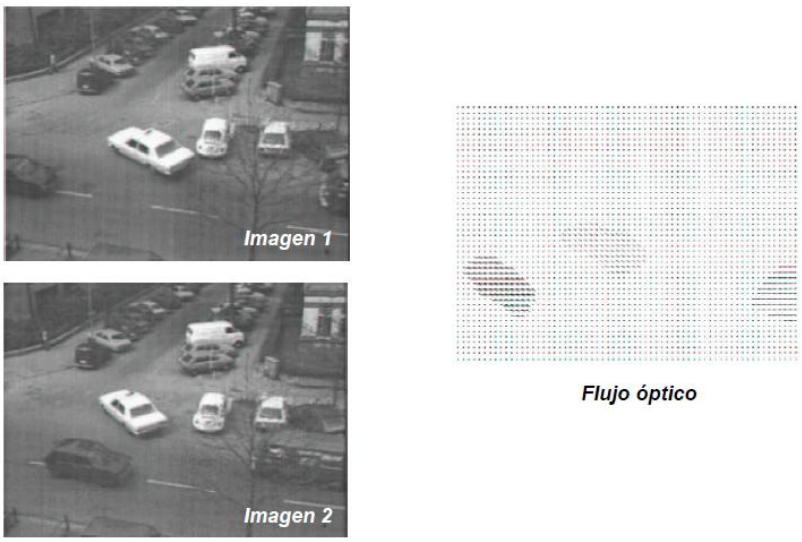



Figura 25. Ejemplo de flujo óptico **[BORDS 00]**.

Se puede pensar entonces, que esta herramienta puede ser de gran utilidad para encontrar la correspondencia entre dos puntos de dos imágenes distintas, ya que para un punto en la imagen $I_1$ se puede buscar su correspondiente en la imagen $I_2$ buscando la intersección del vector de desplazamiento del flujo óptico con la línea epipolar del primer punto en la imagen $I_2$. Existen también técnicas más sofisticadas en que la ventana en la imagen $I_2$ no necesariamente tiene la misma orientación y tamaño de la ventana de la imagen 1 **[OHM 95]**.

Es necesario tomar en cuenta que el flujo óptico presenta dos problemas: oclusión y ambigüedad. En el primero no es posible realizar el matching si el objeto que aparece en la ventana de búsqueda en $I_1$ ya no aparece en la imagen $I_2$. En el problema de ambigüedad, el matching puede equivocarse si lo que se desea buscar en la imagen $I_2$ está repetido. Esta situación se puede producir por ejemplo en la Figura 24 si la ventana de la imagen $I_1$ es más pequeña a la indicada y contiene la ventana izquierda de la torre que aparece en la parte superior derecha de la imagen. Al buscar el matching en la imagen $I_2$ no se podrá diferenciar entre las dos ventanas que tiene la torre, es posible entonces que el algoritmo asocie la ventana izquierda con la ventana derecha.

#### *2.7.2.2 Correspondencia entre líneas*

Establecer la correspondencia entre dos líneas (en dos imágenes distintas) no tiene mucho sentido, ya que en la gran mayoría de los casos existe una línea 3D que podría haber producido las líneas en las imágenes. Tal como se muestra en la Figura 26, la intersección de los planos $(C_1, l_1)$ y $(C_2, l_2)$ proporciona una línea en el espacio 3D, que proyectada en ambas imágenes corresponde con las líneas originales $l_1$ y $l_2$, sin embargo es posible que $l_1$ haya sido producida por otra línea 3D que pertenezca al plano $< C_1, l_1 >$, en este caso la intersección de los planos no proporciona una condición suficiente para establecer la correspondencia. Un caso particular se produce cuando los planos son paralelos y no tienen una línea 3D de intersección, en este caso se dice que las líneas $l_1$ y $l_2$ no son correspondientes. Otro caso particular es cuando ambos planos son iguales, en este caso existen infinitas líneas 3D que podrían

proyectarse como $l_1$ y $l_2$, lo que quiere decir que las líneas podrían ser correspondientes.
Para establecer si dos líneas son correspondientes es conveniente introducir criterios adicionales al descrito anteriormente. Estos criterios deben analizar por ejemplo las características de color de cada una de las líneas o bien la longitud. En el segundo caso sin embargo, es difícil establecer si las líneas en las imágenes son proyecciones de la misma porción de la línea 3D.

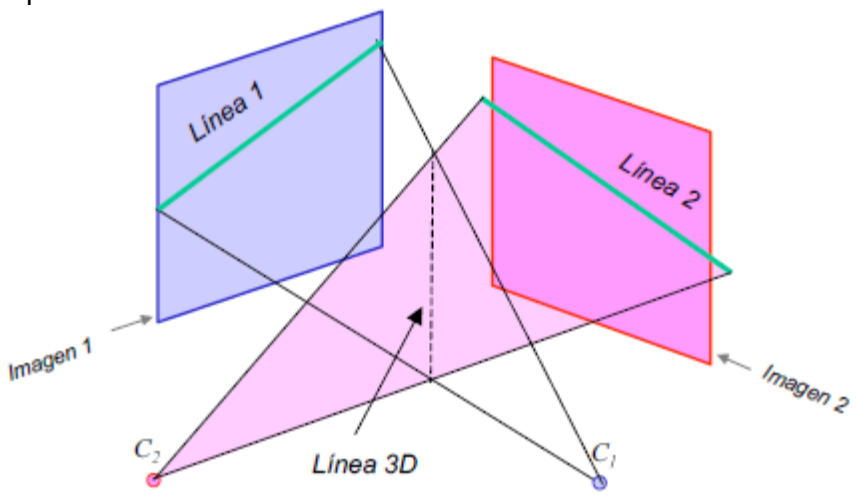



Figura 26. Análisis de líneas correspondientes.

### *2.7.2.3 Correspondencia entre regiones*

Una técnica muy utilizada para establecer la relación de objetos correspondientes en dos imágenes es la de búsqueda de regiones similares. En esta técnica se segmentan en cada una de las dos imágenes las regiones que sean de interés. Luego, se extraen las características de estas regiones y se comparan las características de las regiones de la imagen 1 con las características de las regiones de la imagen 2. Aquellas regiones que tengan características similares y que estén ubicadas obedeciendo la condición epipolar serán entonces regiones correspondientes.

Se entiende por *región* aquel conjunto de píxeles que pertenezcan a una misma zona de la imagen y que esté limitado por bordes. Se asume que los bordes no pertenecen a la región.

Para explicar las características que se detallarán a continuación se usará el ejemplo de la Figura 27. En este ejemplo se presenta una región circular que ha sido segmentada la cual está conformada por los píxeles que pertenecen al círculo (pero no a su perímetro), es decir los píxeles que han sido marcados con color gris en la Figura 27b. Los bordes de la región definen el límite de la región.

Las características que se pueden extraer de una región se dividen en dos categorías: *características geométricas* y *características de color*.

### *2.7.2.4 Características geométricas*

A continuación se enumeran algunas características geométricas que se usan comúnmente en el reconocimiento de patrones.

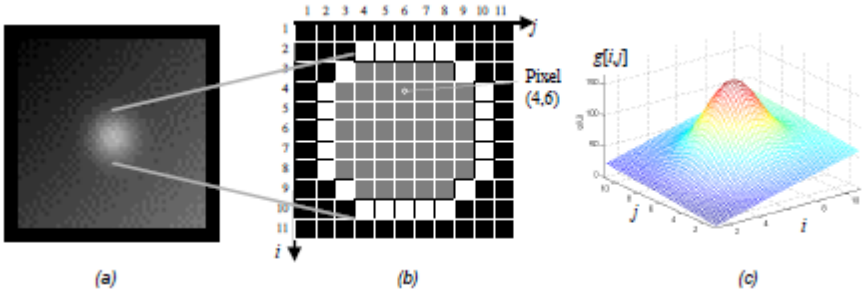



Figura 27. Ejemplo de una región: a) Imagen. b) Región segmentada. c) Representación 3D de los valores de gris de la región y su entorno.

**Altura y ancho (*h* y *w*):** La altura y el ancho de una región se definen como:

$$h = i_{max} - i_{mín} + 1 \qquad \text{y} \qquad w = j_{max} - j_{mín} + 1$$

donde *imax* e *imin* representan el valor máximo y mínimo que toma la coordenada *i* en la región (ver Figura 27), y lo mismo es válido para *jmax* y *jmin.* En el ejemplo mostrado *h* = *w* = 7 píxeles.

**Area (*A*)**: El área de una región se define como el número de los píxeles de la región. En el ejemplo *A* = 45 píxeles.

**Perímetro (*L*):** El perímetro de una región puede ser definido de varias maneras. Una definición práctica, más no exacta, es tomar el perímetro como el número de píxeles que pertenecen al borde de la región 1. En el ejemplo de la Figura 27b, *L* es el número de píxeles marcados en color blanco, es decir
*L* = 24.

**Redondez (*R*):** Esta característica que indica la calidad de redondo de una región es una medida de su forma. La redondez se define como **[HARTMANN 96]**:

$$R = \frac{4A\pi}{L^2}$$

La redondez *R* de una región estará entre los valores 0 y 1. Teóricamente *R* = 1 para un círculo (perfecto); y *R* = 0 para una región que tenga altura y/o ancho igual a cero.

En la práctica sin embargo, debido al muestreo en el espacio de la región estos valores presentan desviaciones como se puede ver en la región circular de nuestro ejemplo. En este caso $R = 4(45)\pi/24^2 = 0,98$.

**Momentos:** Los momentos estadísticos se definen como

$$m_{rs} = \sum_{i,j\ \varepsilon\, \mathfrak{R}} i^r j^s \qquad \text{para r, s } \mathbf{N}$$

donde $\mathfrak{R}$ es el conjunto de píxeles de la región. En el ejemplo de la Figura 27b el pixel cuyas coordenadas son (*i* = 4*; j* = 6) pertenece a este conjunto.

El parámetro *r* +*s* denota el orden del momento. El momento de orden cero $m_{00}$ corresponde al área de la región *A*. El centro de gravedad de una región queda definido por:

$$\bar{\imath} = \frac{m_{10}}{m_{00}} \qquad \bar{\jmath} = \frac{m_{01}}{m_{00}}$$

Con ayuda de las coordenadas del centro de gravedad se definen los momentos centrales que son invariantes al desplazamiento de la región en la imagen.

$$\mu_{rs} = \sum_{i,j\,\varepsilon\Re}(i-\bar{\imath})^r(j-\bar{\jmath})^s \qquad \text{para r, s } \varepsilon \text{ N}$$

Muy conocidos en la teoría de reconocimiento de patrones son las características derivadas de los momentos centrales, denominados momentos de Hu (20, 34):

$$\begin{aligned}
\phi_1 &= \eta_{20} - \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\
&\quad 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}$$

Con

$$\eta_{rs} = \frac{\mu_{rs}}{\mu_{00}^t} \qquad t = \frac{r+s}{2} + 1$$

Los momentos de Hu son invariantes a la traslación, rotación y escalamiento. Esto quiere decir que dos regiones que tengan la misma forma pero que sean de distinto tamaño y que estén ubicados en posiciones y orientaciones distintas en la imagen tendrán momentos de Hu iguales.

A veces sin embargo, es necesario contar con características que además sean invariantes a las transformadas afines. Un conjunto alternativo de características que son invariantes a la traslación, rotación, escalamiento y también a transformaciones afines se puede derivar de los momentos de segundo y tercer orden **[SONKA 98]**:

$$I_1 = \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4}$$

$$I_2 = \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}\mu_{12}^3 + 4\mu_{21}^3\mu_{03} - 3\mu_{21}^2\mu_{12}^3}{\mu_{00}^{10}}$$

$$I_3 = \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7}$$

$$\begin{aligned}
I_4 = (&\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{12}^2 + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} \\
&+ 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} - 8\mu_{11}^3\mu_{30}\mu_{03} \\
&- 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21} + 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} \\
&+ \mu_{02}^3\mu_{30}^2)/\mu_{00}^{11}
\end{aligned}$$

**Descriptores de Fourier:** Una buena caracterización de la forma de una región se logra utilizando los *descriptores de Fourier* **[CHELLAPPA 84], [PERSOON 77], [ZAHN 71]**. Las coordenadas ($i_k$, $j_k$) de los píxeles del borde, para $k$ = 0,…, $L$ - 1 de una región se agrupan en un sentido de giro conformando un número complejo ($i_k$+$j$ .$j_k$) con $j = \sqrt{-1}$, donde $L$ es el perímetro de la región definido como el número de píxeles del borde de la región. La línea continua formada por estas coordenadas corresponde a una señal periódica que puede ser transformada al dominio de Fourier por medio de la Transformada Discreta de Fourier (DFT) **[CASTLEMAN 96]**:

$$F_n = \sum_{k=0}^{L-1}(i_k + j * j_k)e^{-j\frac{2\pi kn}{L}} \quad \text{para n=0,…,L-1}$$

Los descriptores de Fourier corresponden al módulo de los coeficientes complejos de Fourier. Como se puede apreciar los descriptores de Fourier son invariantes a la rotación de la región. El primer descriptor de Fourier $|F_0|$ da información de la ubicación de la región en la imagen. Los descriptores que son invariantes a la posición de la región son los siguientes descriptores.

La fase de los coeficientes de Fourier proporciona información acerca de la orientación y de la simetría de las regiones.

En la Figura 28 se muestran los descriptores de Fourier para el ejemplo de la Figura 27. En este ejemplo el pixel de partida es ($i_0$, $j_0$) = (6, 10). En el caso de un círculo ideal los descriptores serían $|F_n|$ = 0 para 1 < $n$ < $L$, ya que la representación de las coordenadas ($i_k$, $j_k$) corresponderían a una sinusoide perfecta.

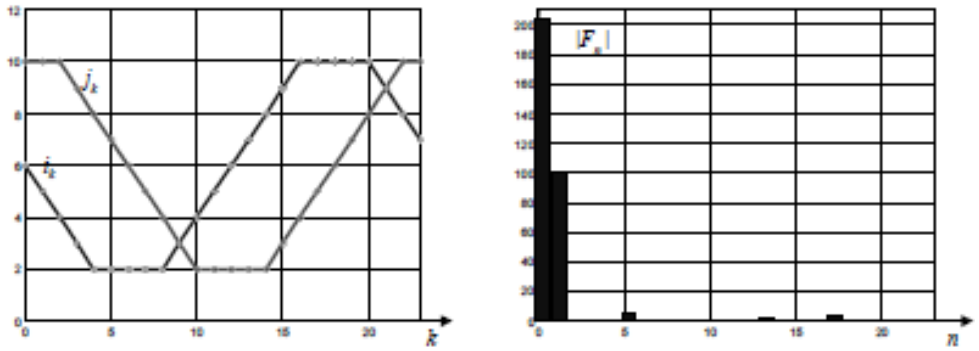


Figura 28. Coordenadas del borde de la región de la Figura 24 y sus descriptores de Fourier.

### *2.7.2.5 Características de color*

Antes de entrar a definir las características del color es necesario saber si la imagen que se pretende analizar es a color o en escala de grises. En el primer caso el color se descompone en tres componentes (rojo, verde y azul) para cada píxel de la imagen, en el segundo caso se cuenta sólo con el tono de gris en cada píxel. Las características que se mencionan a continuación son para una sola variable de color. Esta variable puede ser cada una de las componentes del color, una combinación lineal de las tres componentes o bien simplemente el tono de gris. La información entonces necesaria para calcular estas características es el valor de esta variable de color en cada píxel que es representada como $x[i, j]$ para el píxel ($i$, $j$) de la imagen. Es posible que en algunas aplicaciones sea necesario analizar de manera independiente dos variables de color, por ejemplo la componente en rojo y la componente en azul. En

este tipo de aplicaciones será necesario extraer las características de color para cada una de las variables de color requeridas.

**Color promedio (*G*):** Esta característica es el promedio de la variable de color que se define como:

$$G = \frac{1}{A} \sum_{i,j\ \varepsilon \Re} x[i,j]$$

Donde $\Re$ denota el conjunto de píxeles de la región y *x*[*i, j*] el valor de la variable de color en el píxel (*i, j*).

El número de píxeles de la región $\Re$ es *A*, el área de la región. Una representación 3D de la variable de color de una región y su entorno se muestra en la Figura 27c. En este caso se trabaja con el valor de gris, ya que la imagen está en escala de grises. Para este ejemplo el promedio es *G* = 121.90 (*G*=0 significa 100% negro y *G*=255 corresponde a 100% blanco).

**Gradiente promedio en el borde (*C*):** Esta característica toma el valor promedio del gradiente de la variable de color en el borde de la región. Con esta característica se puede medir qué tan abrupto es el cambio en la coloración en la región con respecto a su entorno. El gradiente promedio en el borde se calcula como:

$$C = \frac{1}{L} \sum_{i,j\ \varepsilon \Re} x'[i,j]$$

donde $x'[i,j]$ es el m´odulo del gradiente de la variable de color del píxel (*i, j*).
Los píxeles a evaluar pertenecen exclusivamente al borde. Estos píxeles conforman el conjunto. El número de píxeles del conjunto es *l*, el perímetro de la región. El gradiente puede ser calculado utilizando el operador de gradiente de Gauss **[CASTLEMAN 96]**, en este caso para el ejemplo de la Figura 25 *C* = 35, 47.

**Promedio de la segunda derivada (*D*):** Esta característica se calcula como el promedio de la segunda derivada de la variable de color en la región:

$$D = \frac{1}{A} \sum_{i,j\ \varepsilon \Re} x''[i,j]$$

donde $x''[i,j]$ denota el módulo de la segunda derivada de la variable de color en el píxel (*i; j*), y $\Re$ el conjunto de píxeles que pertenecen a la región. Para calcular la segunda derivada se puede utilizar el operador LoG (*Laplacian-of-Gauss*) **[CASTLEMAN 96], [MARR 80]**. Es necesario observar que *D* < 0 significa que la región es más clara que su entorno (es decir que su variable de color es mayor en la región que fuera de ella). Así mismo *D* > 0 indica una región más oscura que su entorno.

**Contraste:** El contraste de una región es concebido como una medida para la diferencia de color entre la región y su entorno. 'Región' y 'entorno' no tienen píxeles en común, y conforman una 'zona', que puede ser definida como un rectángulo: La zona entonces queda definida como la ventana

$$g[i,j] = x[i + i_r, j + j_r]$$

para $i$ = 1,…, 2$h$+1 y $j$ = 1,…, 2$w$+1, donde $h$ y $w$ representan la altura y el ancho de la región respectivamente. Los puntos centrales de estas zonas se definen como $i_r = \bar{\imath}$-$h$-1 y $j_r = \bar{\jmath}$-$b$-1, donde $(\bar{\imath}, \bar{\jmath})$ corresponde al centro de gravedad de la región.

Entre más pequeña sea la diferencia de la variable de color en la región con respecto a su entorno, más pequeño será el contraste. Para visualizar el contraste se puede representar la variable de color de una zona como una función 3D, donde el eje $x$ y el eje $y$ representan el eje $i$ y el eje $j$ de la imagen, y el eje $z$ el valor de la variable de color que toma el pixel correspondiente, es decir $g[i, j]$. La Figura 27c muestra esta representación para el ejemplo de la Figura 27a. Se reconoce en este ejemplo una región de alto contraste.

El contraste se define matemáticamente de diversas formas. Una definición comúnmente usada es utilizando características de textura **[CASTLEMAN 96]**. Otras definiciones de contraste **[KAMM 98], [SONKA 98]** se dan a continuación:

$$K_1 = \frac{G - G_e}{G_e} \quad , \quad K_2 = \frac{G - G_e}{G + G_e} \quad , \quad K_3 = ln\left(\frac{G}{G_e}\right)$$

donde $G$ y $G_e$ representan el promedio de la variable de color en la región y en el entorno respectivamente.

Una nueva forma de calcular el contraste se obtiene en tres pasos:

***i)*** **Extracción del color en los ejes principales de la zona:** se calculan dos funciones de color $P_1$ y $P_2$. La primera función $P_1$ toma los valores de la variable de color en la dirección $i$ y la segunda función $P_2$ en la dirección $j$.

Ambas funciones se centran en los centros de gravedad. En el ejemplo de la Figura 29b el centro de gravedad está en el píxel (6,6), esto quiere decir que $P1$ y $P2$ son los valores del tono de gris de la zona de la columna 6 y de la fila 6 respectivamente, tal como se muestra en la Figura 29 a-b.

***ii)*** **Aislamiento de la región:** Para aislar la región de su entorno se trata de eliminar el fondo de la región, que se modela como una función lineal de primer orden, es decir una rampa. Se asume entonces que los valores extremos de $P1$ y $P2$ pertenecen las rampas $R_1$ y $R_2$, tal como se ilustra en la Figura 29 a-b. Las rampas serán substraídas de las funciones originales para conformar $Q_1 = P_1 - R_1$ y $Q_2 = P_2 - R_2$ que se fusionan en la nueva función $Q$ como se muestra en la Figura 29c.

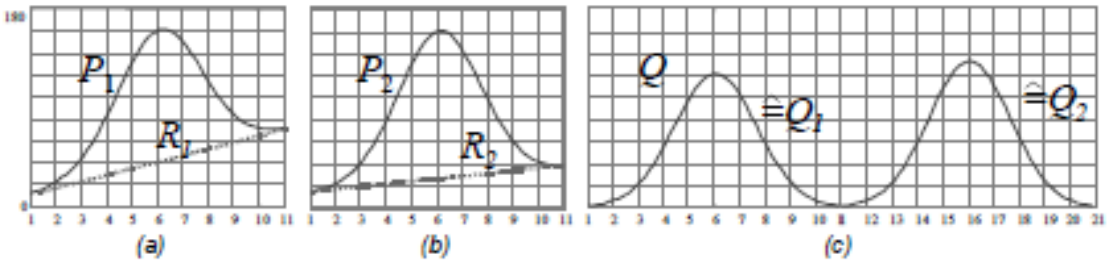



Figura 29. Cálculo del contraste para la Figura 27: a) valor de gris de la zona y rampa en dirección $i$; b) valor de gris de la zona y rampa en dirección $j$; c) fusión de las funciones de a) y b) sin rampas.

***iii)*** **Cálculo del contraste:** a partir de la nueva función *Q* se definen dos nuevos contrastes:

$$K_\sigma = \sigma_Q \qquad K = ln(Q_{max} - Q_{mín})$$

Donde $\sigma_Q$ , $Q_{max}$ y $Q_{min}$ representan la desviación estándar, el máximo y el mínimo de *Q* respectivamente.

**Características de textura:** Las características de textura proporcionan información sobre la distribución espacial del color en la imagen. Para el análisis de regiones se pueden aplicar las características de textura no a la imagen entera sino sólo a las zonas (región y entorno).

Una característica simple de textura es la varianza local **[JAHNE 95]** definida como:

$$\sigma_g^2 = \frac{1}{4hb + 2h + 2b} \sum_{i=1}^{2h+1} \sum_{j=1}^{2b+1} (g[i,j] - \overline{g})^2$$

donde $\overline{g}$ denota el valor promedio de la variable de color en la zona.

Otras características de textura se obtienen por medio de la *matriz de coocurrencia*. La matriz de coocurrencia se denotará como $P_{kl}$, donde el elemento $P_{kl}[i, j]$ otorga el valor de frecuencia (divido por $N_T$ ) de ocurrencia de los valores de color *i* y *j* en dos píxeles ubicados en una posición relativa dada por el vector (*k, l*). La variable $N_T$ significa el número de píxeles que fueron necesarios para calcular P*kl*, con esto se normaliza la matriz de coocurrencia ya que la suma de todos sus elementos es uno.

Si la variable de color tiene una resolución de 256, por ejemplo de 0 a 255, el tamaño de la matriz de coocurrencia $P_{kl}$ será 256*x*256. Ya que esto implica un costo computacional muy alto, es común que se utilicen matrices más pequeñas empleando sólo los bits más significativos de la variable de color **[CASTLEMAN 96]**. A manera de ejemplo, se puede tener una matriz de coocurrencia de 8*x*8 agrupando el valor de la variable de color *x* en [0, ..., 31], [32, ..., 63], ... [224, ..., 255].

Algunas características de textura para imágenes (o zonas) cuyas matrices de coocurrencia sean de $N_x$ *x* $N_x$ elementos se presentan a continuación **[CASTLEMAN 96], [SONKA 98], [HARALICK 79], [FAUGERAS 80]**

**Entropía:**

$$H_{kl} = \sum_{i=1}^{Nx} \sum_{j=1}^{Nx} P_{kl}[i,j] \, log(P_{kl}[i,j])$$

**Inercia o contraste:**

$$I_{kl} = \sum_{i=1}^{N_x} \sum_{j=1}^{Nx} (i-j)^2 \, P_{kl}[i,j]$$

**Homogenidad o energía:**

$$E_{kl} = \sum_{i=1}^{Nx} \sum_{j=1}^{Nx} \left[P_{kl}[i,j]\right]^2$$

**Momento de diferencia inverso:**

$$Z_{kl} = \sum_{i=1}^{Nx} \sum_{j=1}^{Nx} \frac{P_{kl}[i,j]}{1+(i-j)^2}$$

### *2.7.2.6 Criterios de correspondencia entre regiones*

Para establecer si dos regiones $r_1$ y $r_2$, en dos imágenes distintas, son correspondientes se utilizan los siguientes criterios:

- **Condición epipolar:**
  *Los centros de gravedad de las regiones deben satisfacer la condición epipolar*.
  Como la condición epipolar se aplica a puntos (y no a regiones) es necesario simplificar la región en un punto. Esto se hace tomando el centro de gravedad de la región. Esta simplificación está sujeta a un error, que muchas veces puede ser considerablemente grande, ya que los centros de gravedad no necesariamente corresponden a la proyección del mismo punto 3D en el espacio. A manera de ejemplo esta simplificación es cierta si se tratara de regiones esféricas, donde la proyección del centro de gravedad de la esfera coincide con los centros de gravedad de las regiones en las imágenes. Sin embargo, las regiones reales no son proyecciones de esferas, por esta razón el manejo de este criterio debe usarse sabiendo que está sujeto a error. Este criterio sólo debe utilizarse si se trabaja con regiones circulares o si la rotación relativa del objeto (o de las cámaras) en ambas imágenes es pequeña.

  Si $m_1$ y $m_2$ son los centros de gravedad de $r_1$ y $r_2$ respectivamente, es necesario comprobar entonces si satisfacen la condición epipolar. Para esto se utiliza comúnmente la restricción epipolar práctica, que señala que $m_1$ y $m_2$ satisfacen la condición epipolar si $m_2$ se encuentra a una distancia Euclidiana de la línea epipolar de $m_1$ en la segunda imagen a una distancia pequeña.

$$d_2(m_1, F, m_2) = \frac{\left|m_2^T F m_1\right|}{\sqrt{l_1^2 + l_2^2}} < \varepsilon_2$$

  donde F la matriz fundamental existente entre la imagen 1 y 2, y $[l_1 l_2 l_3]^T = Fm_1$

- **Coloración correcta de las regiones con su entorno:**
  *Ambas regiones deben ser o bien más claras, o bien más oscuras que su entorno*.
  Si una región en la imagen 1 es más clara que su entorno se puede pensar que en la imagen 2 su región correspondiente también debería ser más clara que su entorno (suponiendo la misma iluminación). El mismo criterio se puede aplicar para regiones que sean más oscuras que su entorno. Por esta razón, para que las regiones sean correspondientes, la coloración relativa de las regiones con respecto a su entorno debe ser la misma. Para expresar

matemáticamente este criterio se puede decir que el signo del promedio de la segunda derivada de la variable de color de las regiones debe ser igual:

$$sgn(D_1) = sgn(D_2) \quad \textbf{con} \quad sgn(x) \begin{cases} 1\ para\ x > 0 \\ 0\ para\ x = 0 \\ -1\ para\ x < 0 \end{cases}$$

donde $D_1$ y $D_2$ son los promedios de la segunda derivada de las regiones.

- **Criterio de similitud:**
  *Las regiones deben ser 'similares'.*

  Antes de investigar qué tan similares son las regiones, es necesario hacer un estudio de cuales características son las que proporcionan información relevante, esto se logra haciendo un análisis estadístico **[FUKUNAGA 90]**, **[JAIN 00]**.

  Suponiendo que se hayan extraído *n* características de cada una de las regiones $r_1$ y $r_2$ se tiene entonces los vectores de características $z_1 = [z_{11}\ z_{12}\ \dots\ z_{1n}]^T$ y $z_2 = [z_{21}\ z_{22}\ \dots\ z_{2n}]^T$ para $r_1$ y $r_2$ respectivamente. El criterio de similitud evalúa la distancia Euclidiana ente los vectores de características $z_1$ y $z_2$. De esta manera se considera que las regiones son 'similares' si cumplen:

$$S_d(z_1, z_2) = \|z_1 - z_2\| = \sqrt{\sum_{i=1}^{n} (z_{1i} - z_{2i})^2} < \varepsilon_s$$

  donde $\varepsilon_s$ es un valor pequeño.

  Características de fácil computación y que sirven en muchos casos para establecer si dos regiones son similares, son el área (*A*), el perímetro (*L*), el color promedio (*G*) y el contraste (*K*) **[MERY 01]**.

- **Localización correcta en el espacio 3D:**
  *El punto 3D reconstruido, obtenido por triangulación a partir de los centros de gravedad $m_1$ y $m_2$ de las regiones, debe encontrarse dentro del espacio 3D ocupado por el objeto de análisis.*

  Muchas veces se sabe a-priori dónde está ubicado en el espacio el objeto de análisis. Este espacio puede corresponder a un cubo, cilindro o a un volumen más complejo almacenado como un modelo CAD.

  Este criterio corresponde a evaluar la condición epipolar sólo en una porción de la imagen. Esta línea epipolar acotada se obtiene a partir de la proyección en la segunda imagen del segmento de recta $(C_1,m_1)$ que está en el subespacio 3D donde se encuentra el objeto de análisis (ver Figura 30).

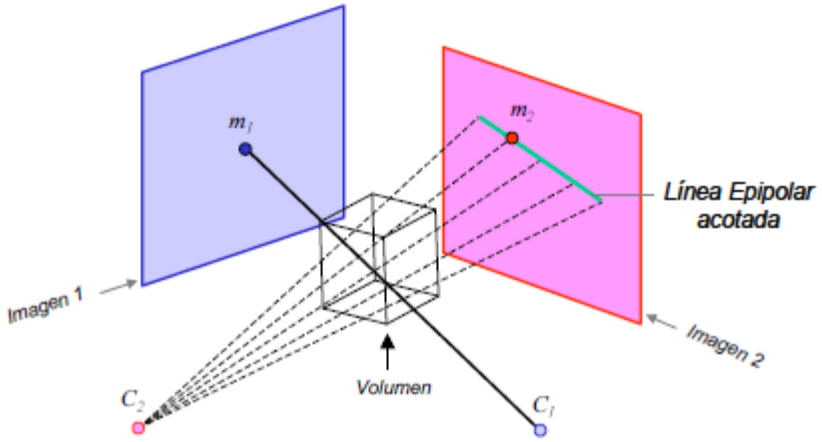



Figura 30. Línea epipolar acotada: la reconstrucción de *M* a partir de $m_1$ a $m_2$ debe pertenecer a un subespacio 3D.

Evaluando estos cuatro criterios se puede establecer con un grado mayor de certeza si las regiones $r_1$ y $r_2$ son correspondientes entre sí.

### 2.7.3 Calibración de cámara

La calibración de la cámara es un proceso que relaciona un modelo ideal con el dispositivo físico y determina la posición y orientación de ésta con respecto al sistema de referencia del mundo.

Dependiendo del modelo utilizado existen diferentes parámetros que tienen que ser determinados.

#### *2.7.3.1 Modelo de cámara pin-hole*

El modelo de cámara de punta de alfiler está basado en el principio de colinealidad, donde cada punto en el espacio del objeto es proyectado en una línea recta a través del centro de proyección del plano de imagen.

La relación entre un punto *M* en 3D y su proyección en la imagen *m* está dado por la fórmula:

$$\boldsymbol{m = A[Rt]M}$$

donde *A* es la matriz intrínseca de la cámara:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{f_x} & \boldsymbol{0} & \boldsymbol{c_x} \\ \boldsymbol{0} & \boldsymbol{f_y} & \boldsymbol{c_y} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

y ($c_x$, $c_y$) son las coordenadas del punto principal;
($f_x$ , $f_y$) son las longitudes focales sobre los ejes *x* y *y*.

- Los parámetros intrínsecos describen la geometría interna y características ópticas de los lentes y del dispositivo de proyección de la imagen.
- La longitud focal es la distancia entre el lente y el plano de imagen.
- El punto principal es el centro de la imagen en coordenadas de píxeles.

(*R*,*t*) son los parámetros extrínsecos que describen la posición y orientación de la cámara en el sistema de referencia del mundo. Relacionan el sistema de coordenadas del mundo con el sistema de coordenadas de la cámara. *R* es la matriz de rotación y *t* el vector de traslación.

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{r_{13}} \\ \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{r_{23}} \\ \boldsymbol{r_{31}} & \boldsymbol{r_{32}} & \boldsymbol{r_{33}} \end{bmatrix}, \boldsymbol{t} = \begin{bmatrix} \boldsymbol{t_1} \\ \boldsymbol{t_2} \\ \boldsymbol{t_3} \end{bmatrix}$$

Un punto en el marco de referencia de la cámara $[X^c, Y^c, Z^z]^T$ está relacionado con su correspondiente punto en el marco de referencia del mundo $[X^W, Y^W, Z^W]^T$ por:

$$\boldsymbol{P^c} = \boldsymbol{R}\boldsymbol{P^W} + \boldsymbol{t}$$

#### *2.7.3.2 Distorsión del lente*

Una cámara generalmente exhibe distorsión producida por el lente. La distorsión es descrita por cuatro coeficientes: dos coeficientes de distorsión radial $k_1$, $k_2$, y dos tangenciales $p_1$, $p_2$.

La distorsión radial provoca que un punto en la imagen sea desplazado radialmente del plano de imagen. La distorsión tangencial es provocada porque no siempre los centros de la curvatura de la superficie del lente son estrictamente colineales.

Si (u, v) son las coordenadas de un píxel de la imagen, es decir, la coordenada de la proyección ideal, y (u', v') es la correspondiente coordenada de la imagen con distorsión y similarmente, (x, y) es la coordenada física ideal (sin distorsión) y (x', y') es la coordenada física real (con distorsión), las coordenadas físicas con distorsión y sin distorsión se relacionan por:

$$x' = x + x[k_1 r^2 + k_2 r^4] + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y' = y + y[k_1 r^2 + k_2 r^4] + [2p_1 xy + p_2(r^2 + 2y^2)]$$

donde $r^2 = x^2 + y^2$.

El segundo término de las ecuaciones anteriores describe la distorsión radial y el tercer término la distorsión tangencial.

Si el centro de la distorsión radial es igual al punto principal (*cx* ,*cy*) entonces:

$$u' = u + (u - c_x)\left[k_1 r^2 + k_2 r^4 + 2p_1 y + p_2\left(\frac{r^2}{x} + 2x\right)\right]$$
$$v' = v + (v - c_x)\left[k_1 r^2 + k_2 r^4 + 2p_2 x + p_1\left(\frac{r^2}{y} + 2y\right)\right]$$

Entre los métodos para calibración de cámaras se destacan el método de Tsai [**TSAI 87**] y el de Heikkila [**HEIKKILA 97**].

## 2.8 CONCLUSIONES

En este capítulo se presentó el estado del arte de los diferentes métodos utilizados para implementar la navegación topológica y de las técnicas de procesamiento digital de imágenes más usadas en este tipo de navegación. Los temas presentados ayudan a esclarecer los procedimientos y algoritmos que generalmente se realizan en cada una de las etapas del procesamiento de imágenes y en la programación de tareas de navegación.

Se selecciona un sistema de navegación topológica relacional debido a que es una de las técnicas más utilizadas, donde se representa el entorno como un grafo o como una red de nodos y aristas. Los nodos representan gateways, landmarks o metas. Las aristas representan una ruta navegable entre dos nodos, los cuales tienen una relación a nivel del espacio.

El entorno en que se llevará a cabo las pruebas de navegación es estructurado donde los objetos presentes en este son estáticos (no cambian de forma ni de posición) y poseen características físicas particulares (forma, color, etc.) que permiten asociarlos con figuras geométricas conocidas como prismas o cilindros o permiten distinguir unos objetos de otros (puertas abiertas, mesas de trabajo, etc.)

Se opta por la utilización de marcas artificiales ya que la detección de estas por medio de una cámara Web es mucho más sencilla, al ser conocido su tamaño, forma y color, permitiendo poder determinar la posición del robot.

Se propone la utilización de una arquitectura de control híbrida ya que amplía las capacidades de operación del sistema de navegación topológico al disponer de una capa reactiva encargada de la evasión de obstáculos en el entorno local y de una capa deliberativa encargada de la planificación y toma de decisiones de un nivel superior como lo es la generación de trayectorias.

De entre los tipos de arquitectura de control híbrida consultados se toman elementos aportados por la arquitectura AURA ya que esta tiene un diseño muy modular con módulos muy bien diferenciados, lo que permite una flexibilidad muy grande a la hora de implementar la Navegación Topológica. Por otra parte, en este tipo de arquitectura la división del control en módulos se realiza por medio del empleo de esquemas reutilizables en la capa reactiva, lo que hace que sea muy adaptable a diversos entornos.

Existen diferentes métodos de planificación de trayectorias para que un robot móvil pueda cumplir la tarea de desplazarse de un punto a otro en un entorno. Para la implementación de la navegación topológica se selecciona el método de planificación de trayectorias por medio de grafos de visibilidad donde se unen cada nodo del mapa topológico mediante un segmento rectilíneo que no intercepta ningún obstáculo.

Se opta por la utilización de un descriptor basado en el algoritmo de campos de potencial apoyado en técnicas reactivas para generar la navegación del robot de un nodo a otro, conociendo con anterioridad la relación que existe entre estos.

La visión artificial en robótica móvil proporciona diferentes algoritmos de procesamiento digital de imágenes como lo son mejoramiento, segmentación y reconocimiento. Dado que en el proyecto se hará uso de una cámara para el reconocimiento de marcas, es importante realizar un proceso de calibración de la

cámara y seleccionar un modelo de color robusto a cambios en la iluminación, que facilite la localización de estas a partir de imágenes capturadas por la cámara.

Para el reconocimiento de imágenes se opta por el algoritmo template matching (comparación de plantillas) donde se tiene una base de datos de las marcas del entorno y se busca en la imagen procesada zonas que tengan correlación alta con esa(s) marca(s).

# CAPITULO III

## 3. ARQUITECTURA HARDWARE, HERRAMIENTAS SOFTWARE Y ENTORNO DE EXPERIMENTACIÓN

Existe mucha variedad de robots: con ruedas, con patas, brazos manipuladores, de forma humanoide, de forma cilíndrica, etc. Sin embargo, la morfología no es una característica esencial, lo que caracteriza a cualquier robot es que combina en la misma plataforma a sensores, actuadores y procesadores. Los sensores miden alguna característica del entorno o propia (por ejemplo cámaras, sensores de obstáculos, etc.). Los actuadores permiten al robot hacer algo, llevar a cabo alguna acción o simplemente desplazarse (por ejemplo los motores). Los procesadores hacen los cómputos necesarios y realizan el enlace lógico entre sensores y actuadores, materializando el comportamiento del robot en el entorno en el cual se encuentra inmerso.

### 3.1 PLATAFORMA HARDWARE

#### 3.1.1 Robot Pioneer 3DX

El *Pioneer 3DX* (mostrado en la Figura 31) es un robot móvil para interiores desarrollado ActivMedia Robotics, empresa que diseña y construye robots móviles inteligentes así como sistemas de navegación, control y de soporte a la percepción sensorial para los mismos.

Figura 31. Robot Pioneer 3DX

El *Pioneer 3DX* es una plataforma robusta que incorpora todos los elementos necesarios para la implementación de un sistema de navegación y control del robot en una gran cantidad de entornos del mundo real. El tamaño del robot es pequeño en comparación con sus prestaciones. Pesa 14 Kg. con una batería y puede transportar hasta 25 Kg. de carga útil. El robot está provisto de un microcontrolador Hitachi HS8 que gobierna diversos dispositivos electrónicos conectados a él (motores, sensores, *encoders*, etc).

En la Figura 32 se muestran los principales componentes del robot.

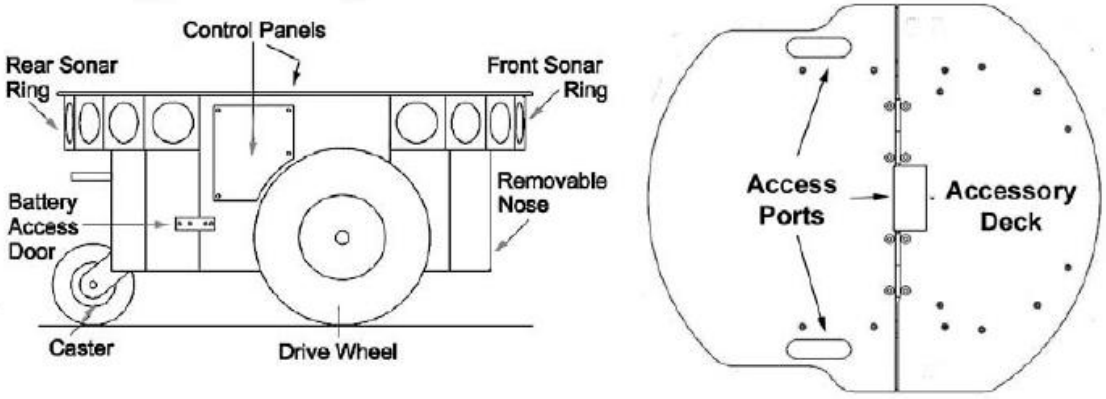



Figura 32. Diferentes Componentes del Robot

***Cubierta Superior:*** No es más que el panel superior del robot. Está destinada al montaje de nuevos accesorios o elementos para el robot, como podrían ser cámaras, láser (SICK) o brazos articulados. La cubierta está dotada con diversos orificios a través de los cuales podrían ubicarse los cables de los posibles dispositivos a añadir. Además a través de ella se puede acceder al interior del robot mediante una ranura de acceso.

**Panel de control:** *Consiste en un panel de acceso al microcontrolador del robot. Está constituido por varios botones de control,* leds *indicadores de estado y un puerto serial RS-232 con un conector DB9.*

En la Figura 33 se muestra el Panel de control del Robot.

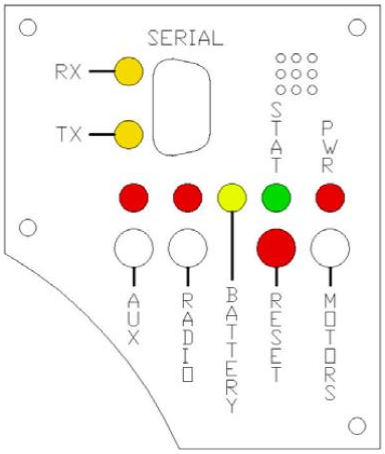



Figura 33. Panel de Control Robot

El led rojo con la etiqueta PWR está encendido siempre que se encienda el robot. El led verde con la etiqueta STAT depende del modo de operación (presenta una lenta intermitencia cuando el microcontrolador espera una conexión con un cliente y rápida cuando el cliente se ha conectado). El led con la etiqueta BATTERY indica el estado de las baterías (rojo si el voltaje está por debajo de 11.5 voltios). El conector serial incorpora dos leds que indican la entrada (RX) y salida (TX) de datos. A través de este puerto se podrá establecer la comunicación con el microcontrolador del robot desde un

PC externo. Este puerto está compartido internamente con el puerto de serie al que se conectará el computador de abordo (*on-board computer*) o bien un radio MODEM. Mediante un circuito electrónico se deshabilita el puerto de serie interno si no existe una conexión con el computador de a bordo o un radio MODEM. AUX1 y AUX2 son dos interruptores que habilitan o deshabilitan respectivamente los puertos de comunicación serial auxiliares. Ambos botones incorporan un led que indica el estado de los mismos. El pulsador RESET resetea el microcontrolador, deshabilitando cualquier conexión activa con éste (sonares, motores, etc). Así mismo existe otro pulsador (MOTORS) que activa o desactiva los motores cuando el robot está conectado a un cliente, y si no lo está simplemente activa el modo auto prueba de los motores.

***Cuerpo del robot:*** El cuerpo del robot mostrado en la Figura 34 está basado en una estructura de aluminio que aloja las baterías, los motores, los circuitos electrónicos y el resto de componentes. Además existe espacio para alojar diversos accesorios, como un *PC on-board*, un radio MODEM o radio *ethernet* o para incorporar sensores adicionales. En la parte inferior exterior del robot se pueden ubicar dos *bumpers* (delantero y trasero) o una pinza robótica.

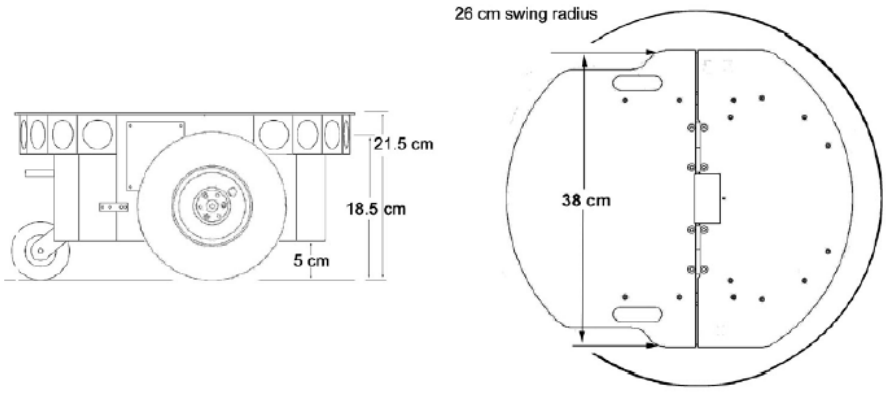



Figura 34. Dimensiones físicas del Robot Pioneer 3DX

***Arreglo de sonares:*** El robot está dotado con dos arreglos de 8 transductores de ultrasonido Polaroid (sonares) cada uno. Estos dispositivos permiten la detección de objetos y la determinación de la distancia a la que se encuentran en referencia al robot. Esta información puede ser utilizada para la elaboración de sistemas de navegación y control. Los sonares están situados en la parte frontal y trasera del robot (opcional). Cada grupo de transductores cubre un rango de 180º, de tal modo que el conjunto de sonares cubre los 360º alrededor del robot.

Cada grupo de sonares tiene su propia tarjeta electrónica controladora, lo que permite realizar un control independiente. Cada grupo de sonares está multiplexado. La frecuencia de adquisición de datos es configurable y viene preestablecida en 25 Hz. (40 milisegundos) por sonar. El rango de detección de objetos va desde los 10cm hasta los 4 metros.

Por software se puede controlar la secuencia de “disparo” de los sonares. Por defecto se disparan del 0 al 7 (arreglo frontal). Igualmente se puede ajustar la sensibilidad de los sonares y el rango detectado mediante un potenciómetro. Esto permite al usuario

adaptar el robot al ambiente que le rodea. Aunque la configuración de los sonares con ganancias bajas reduce sus capacidad de detectar pequeños objetos (lo que puede ser deseable), puede ser beneficioso en el caso de que el robot se mueva en ambientes ruidosos, con superficies muy reflectantes o desiguales. Por el contrario, aumentar la sensibilidad de los sonares estableciendo ganancias altas aumenta las posibilidades de detectar objetos pequeños y objetos que están a gran distancia. Esto es beneficioso si el ambiente en el que opera el robot es abierto, regular y silencioso.

En la Figura 35 se muestra el arreglo de sonares Robot Pioneer 3DX y sistema de coordenadas del robot.

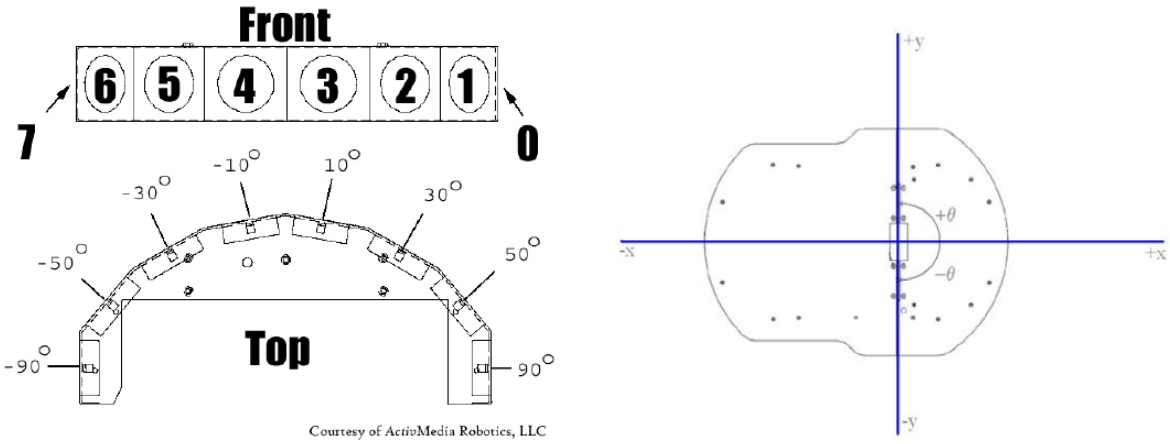



Figura 35. Arreglo de sonares Robot Pioneer 3DX y sistema de coordenadas del robot

***Motores y encoders:*** El robot está dotado de 2 motores de corriente continua reversibles, de alta velocidad y torque. Cada uno de ellos está acompañado de un *encoder* óptico de gran precisión que permiten determinar la velocidad y la posición del robot. La velocidad de los motores es regulada mediante señales del tipo PWM que provienen de controladores PID implementados en el microcontrolador, uno por cada motor. Del mismo modo, este tipo de señales (PWM) son las que transportan la información de los *encoders* hacia el microcontrolador.

***Unidad de Control:*** Los dispositivos electrónicos presentes en el robot y la comunicación hacia el cliente son controlados por un microcontrolador Hitachi HS8/2357. Su función es manejar los actuadores, disparar y recoger la señal de los sónares, controlar la electrónica del robot y realizar el resto de funciones de bajo nivel. Es capaz de comunicarse con otras máquinas a través de una interfaz serial RS232.

La unidad de Control conformada por el microcontrolador y su electrónica asociada, presenta, entre otras, las siguientes características:

- Frecuencia de operación: 18MHz.
- 32K de memoria RAM y 128K de memora FLASH.
- 3 puertos de serie RS-232 configurables entre 9.6 y 115.2 kbaud.
- Entradas para 4 arreglos de sonares con 8 sensores cada uno.
- 2 entradas digitales de 8 bits para los bumpers.
- 1 conector de entrada para pinza robótica con 8 entradas / salidas digitales y una entrada análoga.
- 5 entradas y 2 salidas analógicas.

- 8 entradas y 8 salidas digitales enmascarables.
- Puerto para Joystick de dos botones y dos grados de libertad.
- Salida a buzzer Piezoeléctrico.
- Interfase a tarjeta de control de potencia y motores con líneas PWM de control de dirección del motor y 8 entradas digitales.

***Baterías:*** El robot puede ser alimentado con tres baterías de 12 voltios y 7 amperios-hora cada una. Son intercambiables entre sí y accesibles a través de una mini puerta en la parte trasera del robot.

### 3.1.2 Cámara Logitech Sphere

Para la adquisición de imágenes del entorno se utiliza una cámara web Logitech QuickCam Sphere AF (Figura 36) con movimientos pan/tilt y zoom ubicada sobre el Robot Pioneer 3DX. En la tabla 1 se consignan las características mcon las características más importantes de la cámara.

Figura 36. Cámara Web Logitech QuickCam Sphere AF

Tabla 1. Principales características de la cámara web Logitech QuickCam Sphere AF

| | |
|---|---|
| **Máx. Resolución de Video** | 1600 píxeles x 1200 píxeles |
| **Megapíxeles** | 2 MP |
| **Velocidad** | 30 frames/segundo |
| **Formato video digital** | AVI |
| **Margen de exposición** | 1/10000 seg – 1/5 seg |
| **Imagen fija** | 1280 x 960 |
| **Sensor de imagen** | Tipo CCD |
| **Expansión/Conectividad** | 1xUSB – 4 PIN USB tipo A |
| **Zoom** | 3 x digital |
| **Pan** | +/- 180º |
| **Tilt** | +/- 60º |
| **Peso** | 780 g |

### 3.1.3 Computador HP Pavilon G4

En este computador (Figura 37) se ejecuta el programa cliente que se encarga de enviar peticiones al servidor para el control del robot Pioneer3DX por medio de una conexión remota vía WiFi a través del protocolo de red TCP/IP. Las características de este equipo se consignan en la tabla 2.


Figura 37. Computador HP Pavilon G4

Tabla 2. Principales características del computador HP Pavilon G4

| | |
|---|---|
| **Marca** | HP |
| **Sistema operativo** | Ubuntu 10.04 |
| **Procesador** | Intel® Core(TM) i5-2410M CPU @ 2.30GHz 2.30 GHz |
| **Memoria RAM** | DDR3 SDRAM de 4 GB |
| **Disco duro** | 640 GB SAMSUNG (5400 rpm) |
| **Com. Inalámbrica** | Wireless LAN 802.11 b/g/n WLAN & Bluetooth |

### 3.1.4 Router Wi-Fi LynkSys WRT54G

Para interconectar de forma inalámbrica el computador con el servidor (Pioneer3DX) se emplea el router LynkSys WRT54G (Figura 38).


Figura 38. Router Wi-Fi LynkSys WRT54G

Este hardware cuenta con las siguientes características básicas:

- Switch de 4 puertos Ethernet full duplex 10/100.
- Punto de acceso inalámbrico tipo G (802.11g).
- Tasa de transferencia por conexión inalámbrica de 54 Mbps.
- Compatible con dispositivos Wireless-B.

## 3.2 HERRAMIENTAS SOFTWARE

### 3.2.1 Player [PLAYER/STAGE]

Player es una capa para la abstracción de hardware (*Hardware Abstraction Layer* o HAL) para dispositivos robóticos. Al igual que un sistema operativo oculta los detalles del hardware del ordenador definiendo conceptos genéricos como ratón o impresora,

aportando una interfaz a cada uno de ellos, Player lo realiza con los dispositivos robóticos, por lo que es considerado como un sistema operativo para robots.

Player define una serie de interfaces estándar, cada una de las cuales es una especificación de las formas en las que se puede interactuar con alguna clase de dispositivos. Por ejemplo la interfaz *position2d* es usada por los robots móviles, permitiéndoles aceptar comandos de movimiento (velocidad o posición a alcanzar) y devuelve su estado (velocidad y posición actuales).

El *driver* se encarga de dar soporte a dichas interfaces estándar. Gracias a esto, el código escrito para controlar un robot puede ser utilizado (en cierta medida) para controlar a otros robots.
Aunque la mayoría de los *drivers* de Player controlan directamente el hardware, existen los *drivers* abstractos. Un *driver abstracto* usa otros *drivers*, en lugar de hardware, como fuentes de datos y como destino donde enviar órdenes. El uso fundamental de los *drivers* abstractos es encapsular código para que puedan ser reutilizados fácilmente.

Adicionalmente, Player tiene un diseño cliente/servidor: las aplicaciones establecen un diálogo mediante TCP/IP con el servidor Player [**ESPINOSA 08**], que es el responsable de proporcionar las lecturas sensoriales y llevar a cabo los comandos de actuación. Además de permitir acceso remoto, este diseño proporciona a las aplicaciones construidas sobre PSG (Player/Stage/Gazebo) gran independencia de lenguaje y mínimas imposiciones de arquitectura. La aplicación puede escribirse en cualquier lenguaje, y con cualquier estilo, simplemente respetando el protocolo de comunicaciones con el servidor. El protocolo uniforma el modo en que los programas acceden al hardware. Existen librerías que encapsulan funcionalmente ese diálogo para clientes en varios lenguajes.

Soporta una amplia variedad de dispositivos hardware. Comenzó dando soporte a la serie *Pionner 2* de *ActivMedia Robotics*, pero ha ido ampliando el espectro para proporcionar compatibilidad a dispositivos de uso frecuente en robótica. Su arquitectura modular permite añadir fácilmente soporte para nuevo hardware. Esto le permite disponer de una activa comunidad que contribuye con nuevos *drivers*.

Tanto el código fuente como su documentación están disponibles bajo los términos de la licencia GPL (*General Public License v2*).

Para información sobre los ficheros de configuración de Player, visitar la página Web de Player/Stage sobre configuración **[PLAYER/STAGE]**

### 3.2.2 Librería OpenCV [OpenCV]

La OpenCV implementa una amplia variedad de herramientas o funciones escritas en lenguaje C/C++ para el procesado de imágenes y la visión por ordenador en tiempo real. Si bien existen primitivas orientadas a operaciones de “bajo nivel” en la imágenes, tales como la binarización, el filtrado, las operaciones estadísticas, etc, la OpenCV es fundamentalmente una librería de funciones de “alto nivel” que implementan técnicas y algoritmos de procesado de imágenes tales como la calibración de cámaras, detección de patrones y objetos, “tracking” o seguimiento de objetos, análisis de formas y trayectorias, reconstrucción 3D, segmentación y reconocimiento de objetos, etc, [4].

**La librería de funciones "IPL".** La OpenCV utiliza, o más exactamente incluye, otra librería conocida como IPL. De hecho, las imágenes que se capturan con una cámara, como en nuestro caso utilizando la OpenCV, para ser utilizadas en un determinado sistema de procesado de imagen por ordenador, son convertidas a un formato especifico llamado "IplImage" el cual cuenta con una serie de características y atributos propios definidos por la librería IPL [5]. A diferencia de la OpenCV, la IPL tiene un objetivo más orientado al procesado de las imágenes en "bajo nivel". La IPL no es de libre distribución requiriéndose de una licencia para ser usada.

Ambas librerías han sido diseñadas y optimizadas para los procesadores de arquitectura Intel®, donde se obtienen los mejores resultados de rendimiento y tiempo de ejecución, aunque no dejan de ser compatibles con los demás tipos de procesadores.

Este rendimiento optimizado bajo los procesadores de Intel® permite realizar aplicaciones más rápidas, lo cual en algún tipo de aplicaciones de procesado resulta más que necesario, sobre todo si es una aplicación en la que se requiere un procesado en tiempo real o de alta velocidad, como por ejemplo, la navegación de robot mediante cámaras que se encarguen de detectar los objetos y paredes, para que este pueda sortearlos sin problema antes de llegar al obstáculo.

Ambas librerías presentan compatibilidad absoluta con los sistemas operativos más comúnmente utilizados en ordenadores personales, como son la familia de Microsoft Windows® y la familia de Linux.

### 3.2.3 QT

Qt es un *framework* de desarrollo multiplataforma ampliamente utilizado para el desarrollo de interfaces gráficas de usuario y, desde el lanzamiento de la versión 4, para el desarrollo de aplicaciones no gráficas tales como herramientas de consola o servidores. Sus ejemplos de uso más conocidos son el entorno de escritorio Linux KDE, el navegador web Opera, Google Earth, Skype, etc.

Qt utiliza el estándar de C++, al que añade un pre-procesador que genera el código en C++ necesario para implementar sus propios módulos de extensión.

Uno de los objetivos de diseño más importante de Qt es hacer la programación multiplataforma fácil. Consigue esto abstrayendo la funcionalidad de bajo nivel en las capas inferiores de sistemas de ventanas y sistemas operativos, proporcionando una interfaz coherente y lógica orientada a objetos que ayuda a los programadores.

El API y las herramientas de Qt son consistentes en todas las plataformas soportadas, permitiendo el desarrollo independiente de plataforma y de despliegue. Las aplicaciones de Qt corren de forma nativa, compiladas desde el código fuente, en las siguientes plataformas (Figura 39):

- Qt/Windows: Qt para Microsoft Windows Vista, XP, 2000, NT 4, Me/98/95.
- Qt/X11: Qt para el X Window System (Linux, Solaris, HP-UX, IRIX, AIX, y otras variaciones UNIX).
- Qt/Mac: Qt para Apple Mac OS X.
- Qt/Embedded: Qt para plataformas empotradas (Linux empotrado en PDA, teléfonos móviles, etc.).

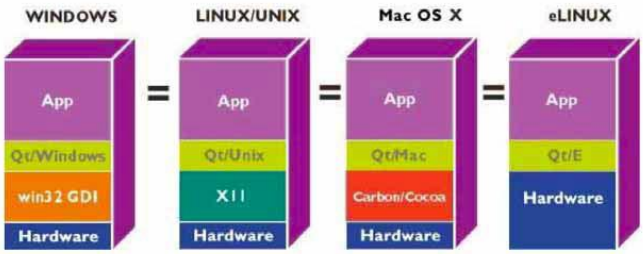



Figura 39. Arquitectura de Qt en las distintas plataformas

La versión libre de Qt utilizada (Qt 3.3.6) para X11 proporciona las siguientes herramientas:

- Biblioteca de clases de Qt: es una biblioteca de cerca de 400 clases (en aumento) que encapsula todas las infraestructuras necesarias para el desarrollo de aplicaciones. La API de Qt incluye un robusto modelo objeto, una gran colección de clases y funcionalidad para el desarrollo de interfaces gráficas de usuario, programación en bases de datos, en redes, multiproceso, XML, internacionalización, integración de OpenGL, etc.

- Diseñador de Qt: es un diseñador de interfaces gráficas de usuario visual e intuitivo, que permite un desarrollo rápido de interfaces de usuario con un *look & feel* nativo sobre todas las plataformas (Figura 40).

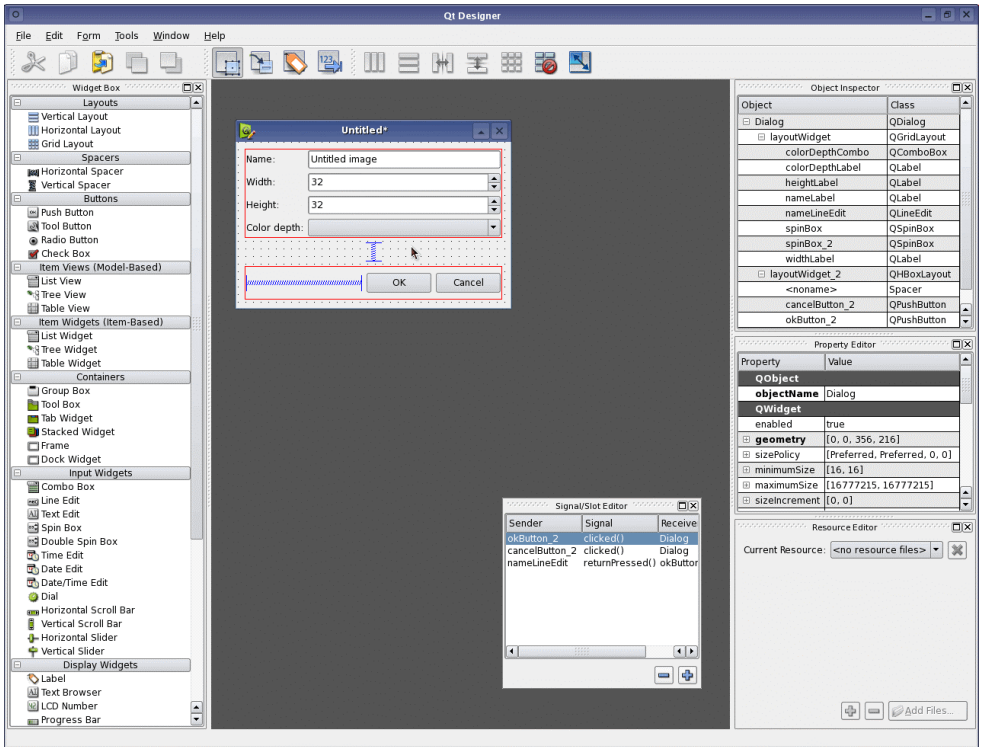



Figura 40. Diseñador de Qt

- Lingüista de Qt: es una serie de herramientas que facilitan la internacionalización de un programa. Estas herramientas permiten que los

desarrolladores deleguen el trabajo de traducción a personal no técnico especializado en traducciones, con lo que se mejora y acelera este proceso.

- Asistente de Qt: es un navegador para documentaciones y archivos de ayuda que puede ser utilizado con cualquier aplicación basada en Qt. Acelera significativamente el proceso de documentación de un proyecto.

## 3.3 SISTEMA OPERATIVO

### 3.3.1 GNU/LINUX

Un sistema operativo generalista que ha ganado gran aceptación en la comunidad robótica es GNU/Linux **[RWI99], [GVH 03], [MRT 03]**. Este sistema es software libre y cuenta con una comunidad muy activa y amplia de desarrolladores, lo cual garantiza en la práctica la incorporación al sistema del soporte de los nuevos dispositivos hardware que van ofreciendo los avances tecnológicos. Este sistema operativo es muy potente y robusto, y ofrece sus servicios en forma de llamadas al sistema y de funciones de biblioteca. Hay llamadas para crear procesos, hebras, controlar su coordinación, comunicaciones, etc. Además incluye un amplio abanico de herramientas de desarrollo, como el compilador de C/C++ gcc, el editor emacs, etc.

Respecto a la multitarea, GNU/Linux incluye la biblioteca Pthreads como referencia para materializar las hebras de las aplicaciones. Este paquete genera hebras POSIX de kernel que se pueden comunicar a través de memoria compartida. GNU/Linux también ofrece semáforos para controlar el acceso a esas variables compartidas o resolver problemas de concurrencia que puedan aparecer. Estos mecanismos se suman a los que ya ofrece GNU/Linux como creación de procesos, tuberías, fifos, memoria mapeada, etc., que siempre están disponibles.

En cuanto a comunicaciones GNU/Linux ofrece los sockets para la comunicación intermáquina entre varios programas, una abstracción que permite olvidarse de los detalles de red, protocolos de acceso al medio, etc. En concreto soporta IP para el nivel de red, y los protocolos del nivel de transporte TCP y UDP. De esta manera el programador de la aplicación no debe preocuparse de localizar la máquina destino, ni modificar su código cuando, por ejemplo, el robot se conecta a la red por ethernet en vez de red inalámbrica.

GNU/Linux ofrece también las librerías necesarias para crear y manejar interfaces gráficas en X-Window, que es el sistema más extendido en el mundo Unix. Encima de las librerías básicas (Xlib y Xt-intrinsics), que son flexibles pero complejas, GNU/Linux dispone de varias librerías que simplifican el manejo de la interfaz gráfica, como Qt, XForms, GTK, etc. Una ventaja natural de las interfaces en X-Window es la visualización remota: es muy sencillo que un programa ejecutándose en un ordenador GNU/Linux (por ejemplo el ordenador a bordo del robot) vuelque su interfaz gráfica a otro ordenador GNU/Linux (por ejemplo el PC de trabajo del programador).

## 3.4 ENTORNO DE EXPERIMENTACIÓN

El entorno de experimentación es un espacio estructurado. Este espacio corresponde a los pasillos del segundo piso del edifico 353 y 354 de la escuela de Ingeniería Eléctrica y Electrónica de la Universidad del Valle. Cierta parte de este espacio presenta una iluminación que puede considerarse estable debido a que proviene de lámparas fluorescentes; La otra parte si presenta muchos cambios de iluminación.

Los obstáculos encontrados en dicho espacio son propios del lugar. En la figura 41 se muestra una fotografía de los lugares donde se llevan a cabo las diferentes pruebas.

Figura 41. Imágenes del Entorno de Experimentación.

## 3.5 CONCLUSIONES

A lo largo del presente capítulo se presentó una descripción de las herramientas hardware y software utilizadas para la implementación del Sistema de Navegación Topológica.

El software OpenCv es una herramienta poderosa y flexible para el tratamiento de imágenes, que al ser un software libre permite a cualquier programador en C/C++ implementar en cualquier entorno de programación programas para una aplicación determinada.

Se escoge un router LynkSys WRT54G de la familia Cisco ya que es un dispositivo robusto que permite interconectar varios computadores mediante enlaces Ethernet 802.3 y 802.11g inlámbricas, y de esta manera garantizar una comunicación estable. Este permite un enlace inalámbrico entre el computador portátil y el robot para ampliar las posibilidades de movilidad de este último.

Gracias a que el Pioneer 3Dx permite una carga de hasta 23 Kg, este se puede equipar con un ordenador portátil y una cámara web para capturar las imágenes del entorno. En el ordenador corre un sistema operativo GNU/Linux, en el cual se va a ejecutar la aplicación de este proyecto. Dispone de una tarjeta de red inalámbrica 802.11 que aporta, al conjunto, la comunicación del robot con otras máquinas.

Las librerías de Qt permiten elaborar una interfaz gráfica la cual facilita la interacción amigable entre el usuario y el sistema implementado en el lenguaje de programación C++.

El entorno de experimentación permite realizar todas las pruebas del sistema de navegación topológica implementado, permitiendo que la plataforma móvil se mueva en un espacio estructurado. Éste cuenta con una iluminación semicontrolada lo que facilita el procesamiento de las imágenes y el reconocimiento de cada uno de los puntos de referencia.

# CAPITULO IV

## 4. SISTEMA DE NAVEGACIÓN TOPOLÓGICA

La Navegación Topológica consiste en que un robot móvil vaya de un lugar a otro mediante indicaciones parecidas a las que se harían a un ser humano que desea realizar el recorrido entre esos mismos lugares.

En este capítulo se presenta el sistema de navegación topológica implementado el cual permite al robot Pioneer-3DX ir de un punto determinado a otro. Estos puntos están representados por marcas artificiales dentro del entorno, las cuales serán detectadas por medio de una cámara web. Para poder llevar a cabo la navegación se tiene un conocimiento previo de la forma de llegar de un punto a otro. En la secuencia detallada de las indicaciones, a la persona se le indica que hacer y hasta cuando tiene que hacerlo. Así, al final se tiene una sucesión de acciones de movimiento que se pretende que guíen al Robot hasta su destino, al igual que se hace con las personas. Esta información, que es de por sí un plan, va a formar parte del mapa topológico considerado (ver Figura 50). Por lo tanto, el mapa va a estar constituido por todas las indicaciones posibles para llegar desde cualquier punto de referencia a otro dentro de un entorno determinado. Más que una sucesión de marcas del entorno, este mapa está formado por una sucesión de tareas a realizar durante la navegación, es decir, todos los planes posibles.

El sistema de navegación topológica implementado se basa en una arquitectura de control híbrida donde la información percibida del entorno de navegación está disponible tanto para el nivel reactivo como para el deliberativo lo que hace posible la construcción de un modelo global orientado a las tareas y una respuesta rápida cuando algún evento lo requiera. Cada uno de los niveles se ejecuta como procesos independientes.

### 4.1 DESCRIPCIÓN DEL SISTEMA

Basado en una arquitectura de control híbrida se implementa el Sistema de Navegación Topológica mostrado en la Figura 42 donde se ilustra un esquema de los módulos de este, dividido en tres subsistemas de la siguiente forma:

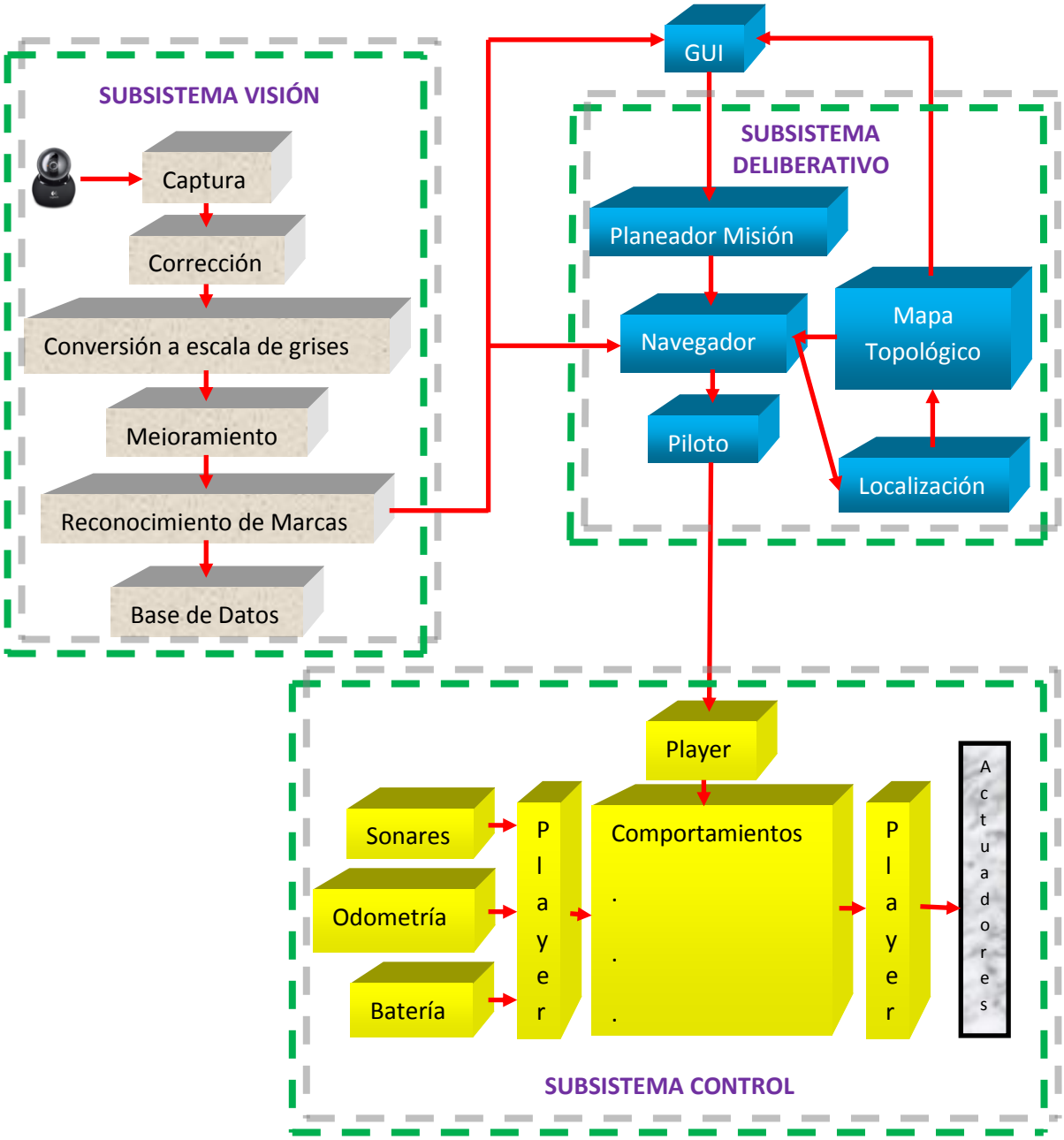



Figura 42. Esquema de los módulos del sistema de Navegación Topológica implementado

### 4.1.1 Módulo Interfaz Gráfica de Usuario (GUI)

Para facilitar la interacción del usuario con el sistema se desarrolla una interfaz gráfica de usuario (GUI) la cual permite visualizar el entorno por donde navega el Robot, las lecturas de los sensores de ultrasonido del robot, su posición con respecto a una posición inicial; y un mapa topológico del entorno donde están ubicados cada uno de los puntos de referencia. Una vez el usuario indique el punto destino desde la interfaz, se indicará en el mismo mapa topológico la ruta a seguir desde la marca donde se encuentre el Robot hasta la marca destino; como también la posición en la que se encuentre el Robot, una vez reconozca la marca. En la Figura 43 se muestra la Interfaz de Usuario que se desarrolló, la cual cuenta con 11 paneles diferentes que se explican brevemente a continuación:

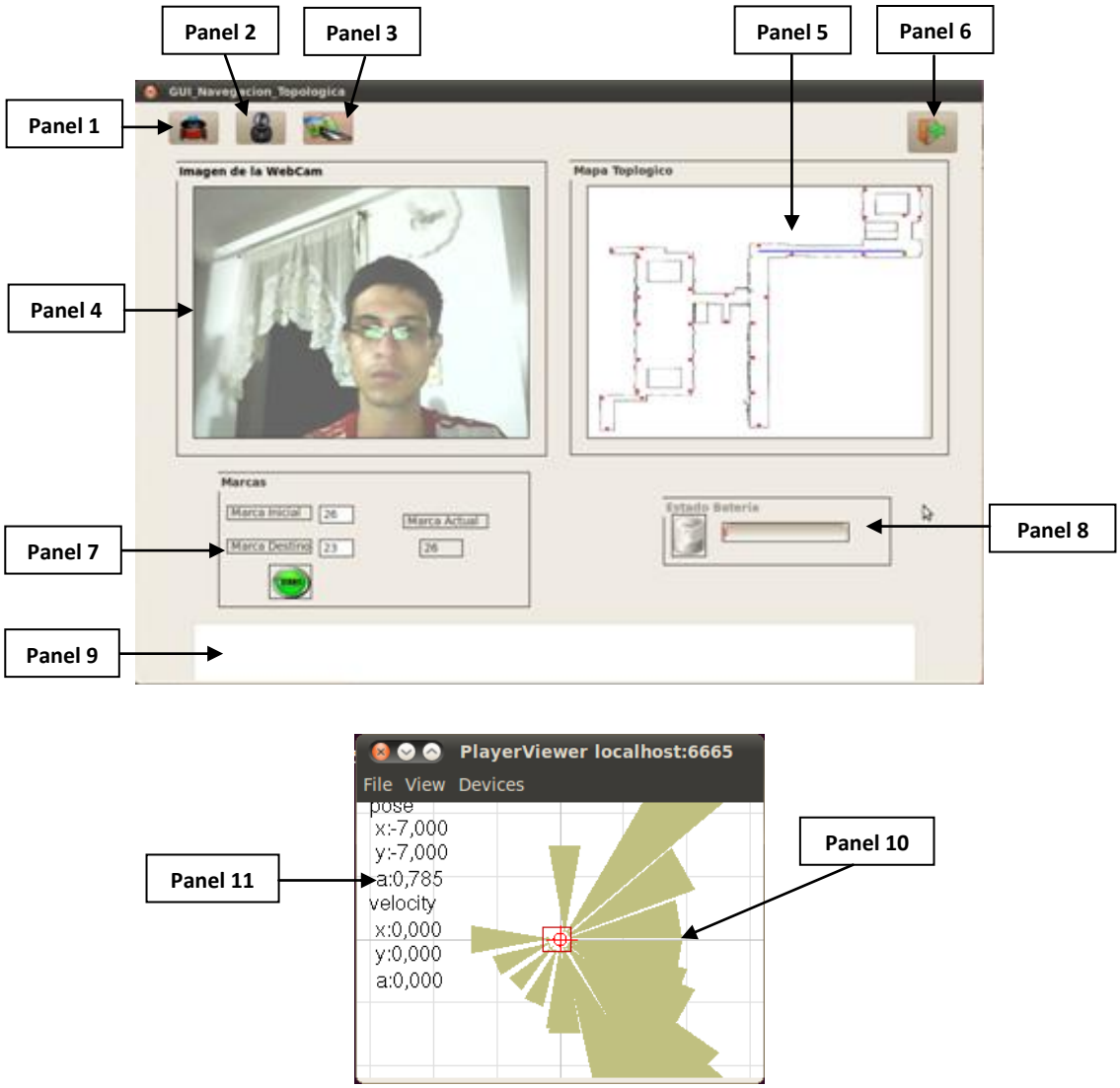



Figura 43. Interfaz Gráfica de Usuario

- ***Panel 1.*** Es el botón que permite conectar el Programa Cliente con el Programa Servidor y así obtener acceso a cada uno de los dispositivos a utilizar del Robot.
- ***Panel 2.*** Es el botón que permite obtener la imagen capturada por la Web Cam e iniciar el procesamiento de la imagen adquirida.
- ***Panel 3.*** Es el botón que permite mostrar el Mapa Topológico del Entorno de Navegación en el Panel 5.
- ***Panel 4.*** Es un área de 600 x 400 píxeles utilizada para mostrar la imagen capturada por la Web Cam.
- ***Panel 5.*** Muestra el Mapa Topológico del Entorno. Cada punto rojo es la posición de cada una de las Marcas de Referencia del Entorno, el punto verde es la ubicación donde se encuentra el Robot cada vez que reconoce una marca por medio de su Sistema de Visión; y las líneas azules representan el camino que debe seguir el Robot desde donde se encuentra hasta la marca destino.
- ***Panel 6.*** Es el botón que permite salir de la aplicación.
- ***Panel 7.*** Permite ingresar el número de la Marca donde se encuentra el Robot inicialmente, el número de la Marca destino donde se desea que el Robot llegue; y muestra el número de la última marca que ha sido reconocida.

- ***Panel 8.*** Muestra en términos de porcentaje el estado en que se encuentra la batería del Robot.
- ***Panel 9.*** Informa la suscripción (acceso) a la plataforma Móvil y a cada uno de los dispositivos de éste.
- ***Panel 10.*** Muestra la lectura de cada uno de los 8 sonares que posee el Robot (esta lectura se da en metros).
- ***Panel 11.*** Muestra las coordenadas en que se encuentra la Plataforma Móvil con respecto a una posición inicial.

### 4.1.2 Módulo Player

Para la implementación del sistema de navegación Topológica se empleo la herramienta software Player/Stage para acceder a los dispositivos del Robot Pioneer-3DX por medio de una conexión remota vía WiFi a través del protocolo de red TCP/IP.

Se implementó un programa servidor que se ejecuta en el Robot y se comunica con su hardware; y un programa cliente en el lenguaje de programación C++, el cual se encarga de enviar peticiones al servidor. Esta comunicación se realiza a través de unas interfaces estándar que proporcionan las librerías de Player. Estas ofrecen un conjunto de operaciones para acceder y controlar los sensores; y permiten controlar un dispositivo a alto nivel y de forma genérica, dejando de lado el lenguaje específico del propio hardware del dispositivo.

Aparte de las interfaces para la comunicación entre el Cliente y el servidor, se utilizan otras para la comunicación con los dispositivos del Robot (sonares, motores, batería). Esto se consigue gracias al uso de los drivers (controladores), que son los encargados de comunicarse directamente con los dispositivos.

Para el caso del Robot Pioneer-3DX se implementaron las interfaces “position2d”, “sonar”, “power”, que se describen brevemente a continuación:

- La interface “position2d” permite controlar las bases del robot móvil en 2D.
- La interface “sonar” proporciona acceso al conjunto de sonares que posee el robot, como una matriz de sonares.
- La interface “power” permite el acceso al subsistema de batería del robot, retornando la lectura de esta en voltios.

#### *4.1.2.1 Programa Servidor*

En Player, la asociación de interfaces/drivers/dispositivos se realiza en el servidor, que es el que se estará ejecutando en el propio robot. Para ello, se debe configurar un fichero, normalmente con extensión “**.cfg**”, en el que se especifican las asociaciones entre drivers e interfaces. Concretamente, se indican los drivers que van a utilizarse y las interfaces que cada uno de éstos drivers utiliza. Asimismo, se indica el dispositivo vinculado a ese driver y esa interfaz.

En el fichero de configuración se especifican bloques de drivers. Dentro de cada bloque, el cual corresponde a la definición de un driver específico, van las siguientes indicaciones:

- El nombre del driver (necesario para que Player lo cargue en ejecución).

- Los dispositivos que el driver ofrece. Para cada dispositivo, se indica la interfaz que va a utilizarse para comunicarse con él, y el índice del dispositivo (ya que puede haber más de un sensor del mismo tipo).
- El puerto a través del cual el driver se va a comunicar con los dispositivos.
- Otras opciones más avanzadas, las cuales dependen del driver a utilizar.

La estructura de un fichero de configuración de Player, define una sección por cada driver que se vaya a utilizar. Para cada sección hay que especificar las características de él y los dispositivos que va a controlar.

Los ficheros de configuración de Player poseen una sintaxis clara y definida, que permite definir los diferentes aspectos de los drivers y los dispositivos que éstos controlan. Cada sección viene definida por la palabra clave “driver”, seguido de unos paréntesis. Dentro de éstos se definen las opciones de cada uno mediante pares de “opción-valor”, los cuales se separan por espacios. Cada driver tiene unas opciones de configuración concretas, aunque existen otras independientes de cualquiera de ellos. A continuación se explican las opciones generales, las cuáles son válidas para todos:

- ***name*** (string): Es el nombre del driver que se va a instanciar y es obligatorio.

- ***plugin*** (string): El nombre de la librería compartida que contiene el driver. Esta se intentará cargar antes de instanciarlo. Se utiliza para drivers no incluidos por defecto en Player.

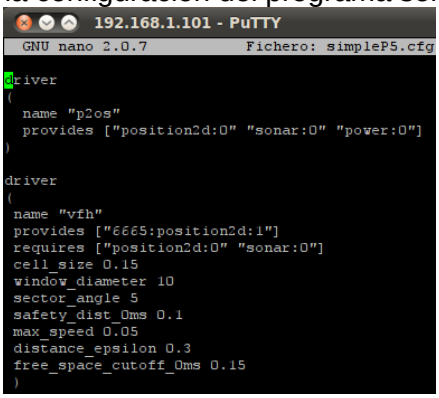
- ***provides*** (tupla de strings): Los dispositivos a los que se tendrá acceso a través del driver. Dado que cada uno puede ofrecer varios dispositivos, éstos se especifican entre corchetes y separados por un espacio. Su uso es obligatorio.

- ***requires*** (tupla de strings): Los dispositivos a los que el driver va a suscribirse, los cuales son accesibles a través de otro driver el cual ha de declararse en primer lugar.

- ***alwayson*** (int): Si es 1, el driver se instancia al arrancar Player, sin esperar a que un programa cliente se conecte.

Aparte de estas opciones generales e independientes, cada uno puede tener otras (para indicar el puerto de conexión, etc.), pero éstas son específicas de cada driver y es el desarrollador de éste quien las especifica.

A continuación se muestra la configuración del programa servidor implementado:



***VFH*** (*VectorField Histogram*), creado por Ulrich y Borenstein **[IWAN 98]**, es un algoritmo muy conocido y utilizado para la generación de trayectorias y el replanteamiento de éstas, y también para la evasión de obstáculos mientras el robot se dirige a su destino.

Este algoritmo se basa en el uso de un histograma cartesiano de dos dimensiones para representar obstáculos. En cada celda de éste hay un valor que representa la certeza de que exista un obstáculo en esa posición. Para ello, este algoritmo necesita trazar una rejilla en el espacio que capta con su sistema sensor y no usa un mapa, por lo que la rejilla que traza es local.

El funcionamiento de este algoritmo puede resumirse en los siguientes pasos:

- El entorno local del robot se representa en una rejilla de dos dimensiones, en la que los valores de las celdas almacenan la probabilidad de contener un obstáculo, calculada a partir de las mediciones realizadas por el sistema sensor.
- Se reduce el histograma cartesiano a un histograma polar de una dimensión. De este modo se obtiene la probabilidad de encontrar un obstáculo en las direcciones de giro.
- Se buscan todas las rutas disponibles al destino, y se selecciona el que menor función de coste G tenga.

$$G = a \cdot TD + b \cdot WO + c \cdot PD$$

Siendo:

- **TD (*target direction*):** alineación del robot con respecto al objetivo WO (*wheel orientation*): diferencia entre la nueva dirección y la orientación actual de las ruedas.
- **PD (*previous direction*):** diferencia entre la dirección previamente seleccionada y la nueva dirección.
- **a, b, c:** parámetros de ajuste del comportamiento del robot.

Es decir, se busca la dirección de avance ponderando la información que se posee de la dirección actual y la dirección de destino.

VFH emplea una técnica de reducción de datos en dos fases, transformando así los datos en los tres niveles que se explican a continuación:

- En el primer nivel se encuentra la “descripción detallada” del entorno del robot. En este nivel el histograma cartesiano de dos dimensiones se actualiza continuamente en tiempo real mediante la información proveniente de los sensores de a bordo del robot.

- El nivel intermedio de datos lo construye un “histograma polar” de una dimensión alrededor de la posición instantánea del robot. El histograma contendrá una serie de sectores angulares (*k*) de una anchura determinada. La transformación del “entorno detallado”, descrito en el primer nivel, al histograma mencionado, resulta de colocar en cada sector *k* un valor que represente la densidad polar de obstáculos en la dirección correspondiente a dicho sector angular.

En la Figura 44 aparece representada la transformación de la descripción detallada del entorno en el histograma polar.

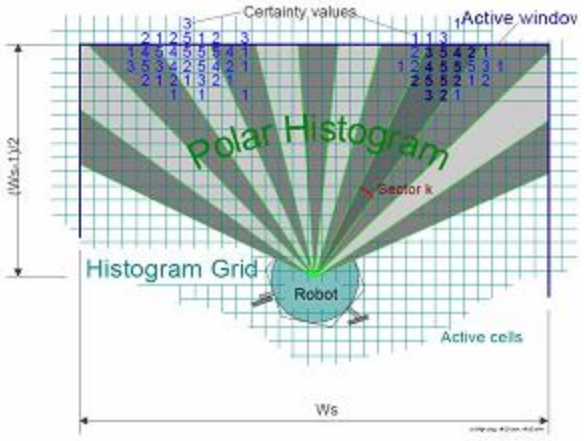



Figura 44. Transformación del primer nivel de datos a histograma polar en el algoritmo VFH **[BORENSTEIN 91]**

En la Figura 45 aparece representado el nivel intermedio de datos. Y en la Figura 46 aparece el diagrama que representa la segunda transformación de datos. En él, aparecen en color verdes los sectores por donde es seguro que el robot circule, y en rojo los que no lo son.

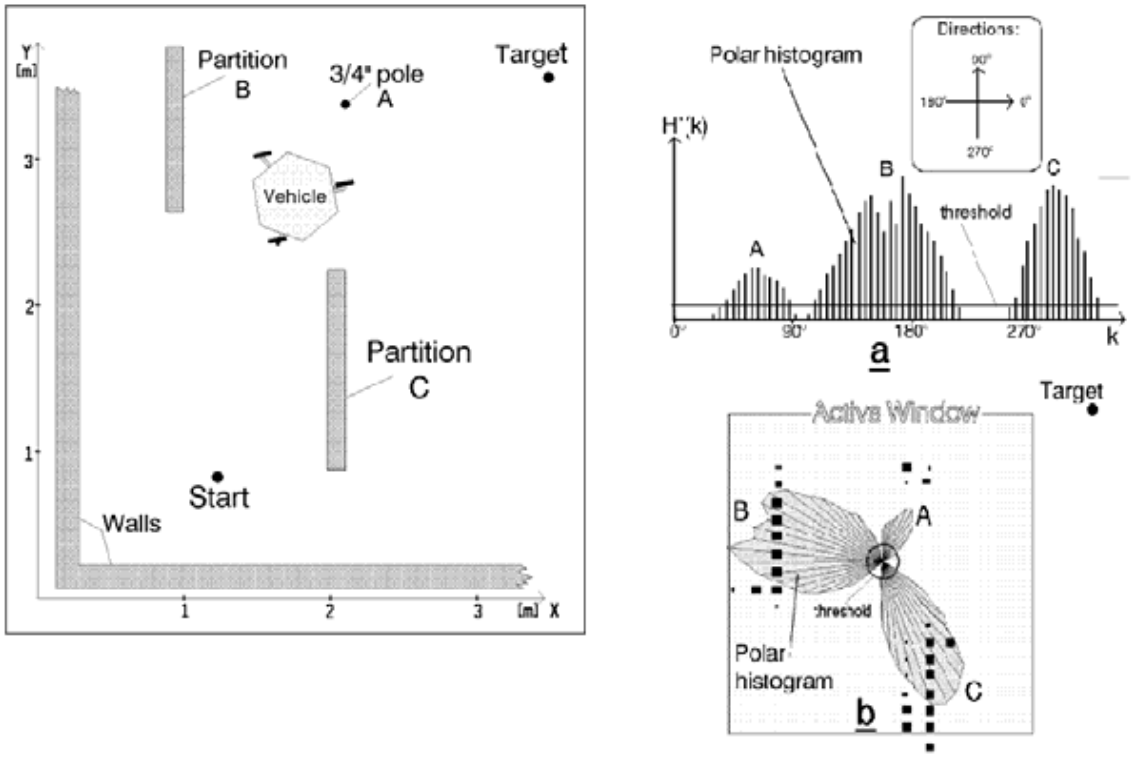



Figura 45. Nivel intermedio de datos en el algoritmo VFH **[BORENSTEIN 91]**

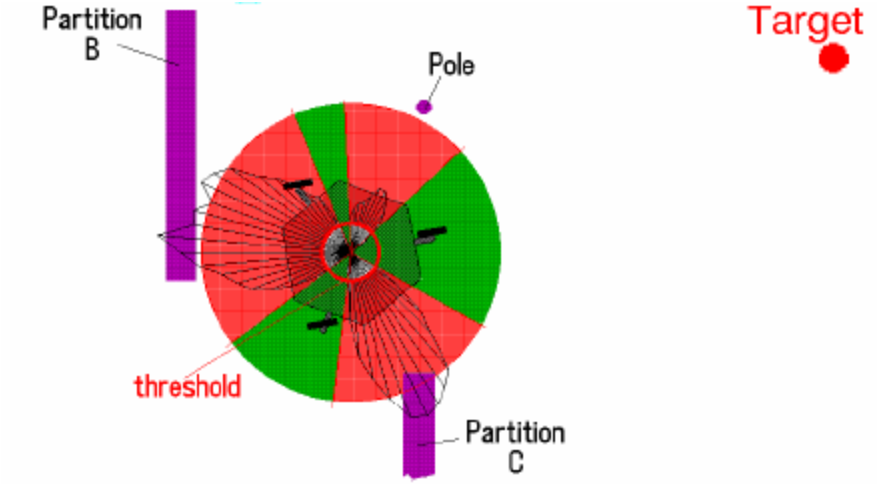



Figura 46. Diagrama de sectores seguros y peligrosos en el algoritmo VFH **[BORENSTEIN 91]**

- El último nivel de la representación de datos es la salida del algoritmo VFH: los valores de referencia para la conducción del vehículo.

El algoritmo VFH está implementado en Player donde se comporta como driver que requiere dos dispositivos y provee uno, como a continuación se especifica:

- Un dispositivo de posición (*position2d*).
- Un dispositivo sensor, ya sea láser o de ultrasonidos (*laser* o *sonar*).
- Provee otro dispositivo de posición (*position2d*).

Este driver posee una serie de parámetros para adaptarlos a nuestras necesidades, los cuales se detallan a continuación:

- ***cell_size*:** tamaño de las celda del mapa local de ocupación. Valor por defecto: 0,1m.
- ***window_diameter*:** fija la dimensión del mapa de ocupación (el mapa consiste de *window_diameter* x *window_diameter* celdas). Valor por defecto: 61. Por tanto, las dimensiones del mapa serán por defecto 6,1 m x 6,1 m (61 x 61 x 0,1 m).
- ***sector_angle*:** resolución angular del histograma polar en grados. Valor por defecto: 5º.
- ***safety_dist_0ms*:** distancia mínima a la que se permite al robot acercarse a los obstáculos cuando está parado. Valor por defecto: 0,1m.
- ***safety_dist_1ms*:** distancia mínima a la que se permite al robot acercarse a los obstáculos cuando lleva una velocidad de 1m/s. Valor por defecto: *safety_dist_0ms*.
- ***max_speed*:** máxima velocidad permitida para el robot. Valor por defecto: 0,2m/s.
- ***max_speed_narrow_opening*:** máxima velocidad permitida al robot en espacios estrechos. Valor por defecto: *max_speed*.
- ***max_speed_wide_opening*:** máxima velocidad permitida al robot en espacios anchos. Valor por defecto: *max_speed*.
- ***max_acceleration*:** máxima aceleración permitida al robot. Valor por defecto: 0,2m/s2.

- ***min_turnate*:** mínima velocidad de giro permitida al robot. Valor por defecto: 10º/s.
- ***max_turnate_0ms*:** máxima velocidad de giro permitida al robot cuando está parado. Valor por defecto: 40º/s.
- ***max_turnate_1ms*:** máxima velocidad de giro permitida al robot cuando va a una velocidad de 1m/s. Valor por defecto: *max_turnate_0ms*.
- ***min_turn_radius_safety_factor*:** este parámetro del driver aun no está explicado en la ayuda, por lo que su uso no queda claro en este proyecto. Valor por defecto: 1.0.
- ***free_space_cutoff_0ms*:** parámetro adimensional. Cuanto mayor es el valor de este parámetro más se aproxima el robot a los obstáculos antes de evitarlos, mientras está parado. Valor por defecto: 2000000,0.
- ***free_space_cutoff_1ms*:** parámetro adimensional. Cuanto mayor es el valor de este parámetro más se aproxima el robot a los obstáculos antes de evitarlos, mientras va a una velocidad de 1m/s. Valor por defecto: *free_space_cutoff_0ms*.
- ***obs_cutoff_0ms*:** este parámetro del driver aun no está explicado en la ayuda, por lo que su uso no queda claro en este proyecto. Valor por defecto: *free_space_cutoff_0ms*.
- ***obs_cutoff_1ms*:** este parámetro del driver aun no está explicado en la ayuda, por lo que su uso no queda claro en este proyecto. Valor por defecto: *free_space_cutoff_0ms*.
- ***weight_desired_dir*:** error de ángulo permitido al robot para dirigirse al destino sin replantear el trayecto. Valor por defecto: 5,0.
- ***weight_current_dir*:** error de distancia permitido al robot para continuar dirigiéndose hacia el destino. Valor por defecto: 3,0.
- ***distance_epsilon*:** distancia de error alrededor del destino aceptada para considerar que el robot ha llegado a éste. Valor por defecto: 0,5m.
- ***angle_epsilon*:** ángulo diferencia entre la orientación del robot y el ángulo se le ha pedido a éste, que se considera aceptable para considerar que ha alcanzado la orientación correcta. Valor por defecto: 10º.

Este driver incluye unas opciones para el escape en caso de parada por choque. En caso de que el dispositivo de posición del robot devuelva una señal de ***stall*** (indicador de choque del robot), el algoritmo iniciaría un procedimiento de escape. Este procedimiento hace que el robot se mueva, durante un determinado tiempo, hacia delante o hacia atrás mientras gira. Los parámetros que controlan este procedimiento de escape son los siguientes:

- ***escape_speed:*** si no es cero, indica la velocidad que se usa mientras se intenta escapar. Valor por defecto: 0,0m/s.
- ***escape_time:*** si no es cero, indica el tiempo (en segundos) que durará el intento de escape. Valor por defecto: 0,0s.
- ***escape_max_turnate:*** si no es cero, representa la máxima velocidad angular utilizada mientras se intenta escapar. Valor por defecto: 0,0º/s.

#### *4.1.2.2 Programa Cliente*

Para el desarrollo del programa Cliente se utilizaron las librerías que Player proporciona y se hizo uso de las interfaces para controlar el robot. A través de las interfaces, se pueden leer los datos provenientes de los sensores, así como enviar comandos a los mismos.

La estructura del programa cliente desarrollado, utilizando las librerías de Player, es la siguiente:

- Conexión del cliente con el robot (con el servidor).

- Acceso a los dispositivos o sensores utilizando las interfaces apropiadas. En esta fase se realiza lo que se llama suscripción a las interfaces, que no es otra cosa más que vincular un objeto "proxy" con un dispositivo o sensor. Existe un "proxy" para cada tipo de sensor y son representados por una clase con sus atributos y sus métodos, de forma que trabajar con los sensores es mucho más sencillo y flexible.

- Creación de un bucle. Este es infinito y representa el ciclo de vida del programa.

- Lectura de los datos de los sensores (Cámara Web, odometría, sonares, batería). Se hace una vez por iteración.

- Procesamiento de la Imagen adquirida y realización de las acciones del movimiento del robot. Más adelante se hablará más detalladamente en la explicación de los siguientes módulos del sistema.

- Cierre de la conexión y salir del programa.

En pseudocódigo, la estructura del programa Cliente es de la siguiente forma:

```
conectarRobot(IP, puerto)
conectarDispositivos (robot, índice)
while (condicionParada)
{
leerDatos()
logicaPrograma()
enviarComandos()
}
desconectarDispositivos()
desconectarRobot()
```

A continuación se muestra parte del programa cliente desarrollado en el lenguaje de programación C++:

En dicho programa se incluyen primero las librerías necesarias que proporcionan las funciones que se van a utilizar. Así, se emplean las librerías de Player para C++, incluidas en la librería padre "libplayerc++"; y las de OpenCV para el tratamiento digital de las imágenes.

```
#include <libplayerc++/playerc++.h>
#include "cv.h"
#include "highgui.h"
#include <string.h>
#include "cvaux.h"
#include "cxcore.h"
```

Seguidamente se realiza la conexión con el robot. Para ello se crea un cliente indicando su dirección IP, la cual para el robot Pioneer-3DX es “192.168.54.101”; y el puerto por el cual se va acceder al robot (6665).

- `client = playerc_client_create(NULL,"192.168.1.101",6665);`

Para cerciorarse de que la conexión fue un éxito, se establece la siguiente condición:

- ```
  if (playerc_client_connect(client) != 0)
  return -1;
  ```

Si la conexión fue satisfactoria se devolverá un 0 notificando al servidor que un nuevo cliente desea recibir datos, de lo contrario se devolverá un 1 y se cerrará el programa informando que hay un error.

Una vez que se realiza la conexión con el robot, se accede a los dispositivos (sonar, motor, batería), suscribiéndose a ellos a través de un objeto “proxy”, que tal y como se ha mencionado con anterioridad, representan cada uno de los dispositivos, encapsulándolos en un objeto con sus atributos y métodos específicos. En este caso, se dispone de un dispositivo sonar, un motor, y una batería (accesible a través de la interfaz position2d). Para acceder a estos últimos, es necesario suscribirse a ellos, por lo que se realizan llamadas a los constructores de los “proxys”, indicando el cliente con el que se ha establecido la conexión y el índice del dispositivo (por si hay más de un dispositivo del mismo tipo).

- `sonar = playerc_sonar_create(client, 0);`
- `position2d = playerc_position2d_create(client, 1);`
- `power = playerc_power_create(client, 0);`

La función ***create*** crea un proxy del nuevo dispositivo y devuelve un puntero para ser utilizado en las llamadas a funciones en futuro.

Luego de haber creado cada uno de los dispositivos hay que verificar que la suscripción a cada uno se realizó satisfactoriamente por medio de las siguientes condiciones, donde cada una se refiere a los dispositivos ya mencionados:

- ```
  if (playerc_sonar_subscribe(sonar, PLAYERC_MODE_ALL)!=0)
  return -1;
  ```
- ```
  if(playerc_position2d_subscribe(position2d,PLAYERC_MODE_AL
  L)!=0)
  return -1;
  ```
- ```
  if (playerc_power_subscribe(power, PLAYERC_MODE_ALL) !=0)
  return -1;
  ```

La función ***subscribe*** notifica al servidor que el cliente está utilizando el dispositivo (sonar, posición, batería), y que el cliente espera enviar y recibir datos (PLAYER_MODE_ALL).

Después de especificar la conexión y de suscribirse a los dispositivos necesarios, se inicia el bucle principal, que en este caso es infinito.

En cada una de las iteraciones se leen los datos de **TODOS** los dispositivos. Una única llamada al robot, permite que éstos los retornen sobre los objetos "proxy". De este modo, con esta llamada se puede acceder a los objetos vinculados con los dispositivos para obtener sus datos. La llamada de lectura de éstos es la siguiente:

- `playerc_client_read(client);`

La llamada a "`playerc_client_read()`" provoca que se actualicen, eliminando los anteriores. No es una llamada bloqueante, por lo que si algún sensor no dispone de nuevos, no espera por ellos, sino que devuelve los que tenga en ese momento, aunque sean antiguos.

Una vez que se han leído los datos de los sensores, se puede hacer cualquier análisis u operación. En este caso, se analizan los provenientes del sensor sonar y si se detecta que el robot tiene un obstáculo delante, se envían instrucciones al motor para que lo evada de acuerdo a los parámetros establecidos en el driver VFH. También se analizan aquellos de la odometría para saber la posición del robot con respecto a un punto inicial; y la lectura de la carga de la batería en voltios.

El envío de nuevas instrucciones al actuador del motor, se realiza a través de la interfaz "position2d" del siguiente modo:

- `playerc_position2d_enable(position2d, 1);`
- `playerc_position2d_set_odom(position2d, 0, 0, 0);`
- `playerc_position2d_set_cmd_pose_with_vel(position2d, pos_plan, vel, 1);`

La función ***playerc_position2d_enable*** envía una solicitud de configuración al servidor, cambiando el estado del motor del robot de apagado a encendido, permitiendo que éste se mueva.

La función ***playerc_position2d_set_odom*** establece la coordenada y la dirección en que inicia la odometría.

La función ***playerc_position2d_set_cmd_pose_with_vel*** establece la posición a la cual se desea que la plataforma móvil llegue a una velocidad dada.

Para dar de baja y destruir el proxy de los dispositivos se usan las siguientes funciones:

- `playerc_position2d_unsubscribe(position2d);`
- `playerc_sonar_unsubscribe(sonar);`
- `playerc_power_unsubscribe(power);`
- `playerc_position2d_destroy(position2d);`
- `playerc_sonar_destroy(sonar);`
- `playerc_power_destroy(power);`

La función ***unsubscribe*** le dice al servidor Player que el cliente ya no está usando el dispositivo.

La función ***destroy*** libera la memoria asociada con el proxy del dispositivo; el puntero del dispositivo estará ahora inválido y no deberá volverse a utilizar.

Para desconectar y destruir el proxy del cliente se utilizan las siguientes funciones:

- `player_client_disconnect(client);`
- `player_client_destroy(cliente);`

La función ***disconnect*** indica al servidor que el cliente se está cerrando.

La función ***destroy*** libera la memoria asociada con el proxy de cliente; el puntero del cliente estará ahora inválido y no debe volver a utilizarse.

En la Figura 47 se muestra el esquema de conexión entre el Robot y el PC; y en la Figura 48 el esquema de acceso al robot "interfaces/drivers/dispositivos":

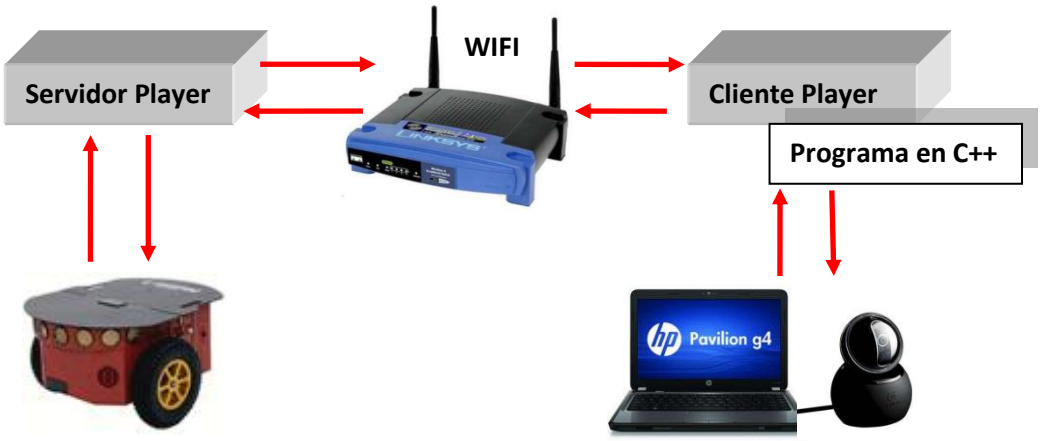



Figura 47. Esquema de conexión entre el Robot y el PC.

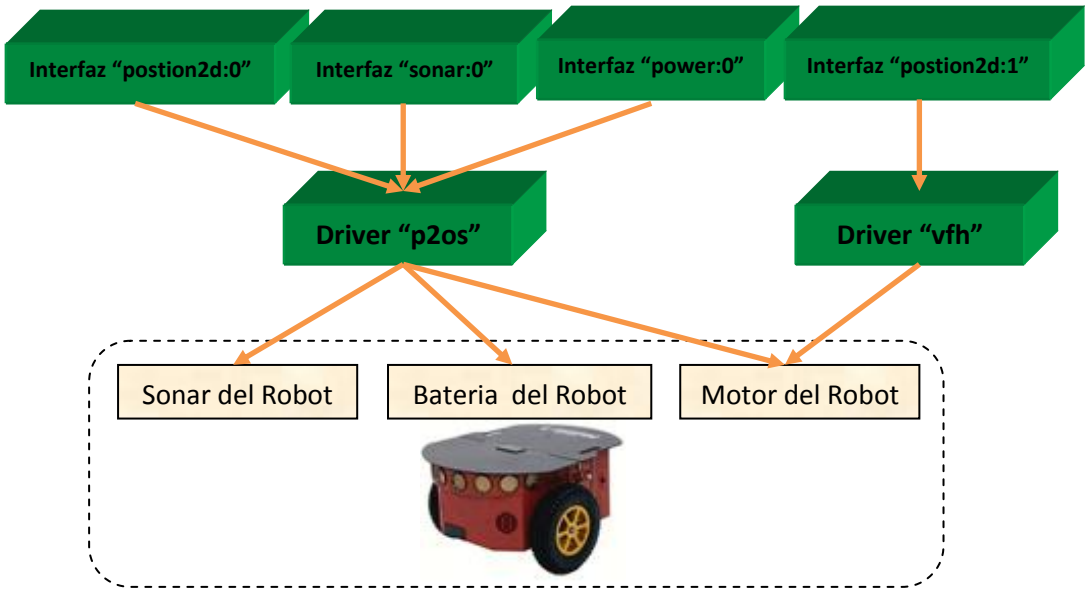



Figura 48. Esquema de acceso al robot "interfaces/drivers/dispositivos"

Antes de inicializar el sistema se debe posicionar el Robot frente a una de las marcas de referencias del entorno. Esta posición se toma como la posición inicial con coordenadas (0,0) a una orientación de 0° como se muestra en la Figura 49.

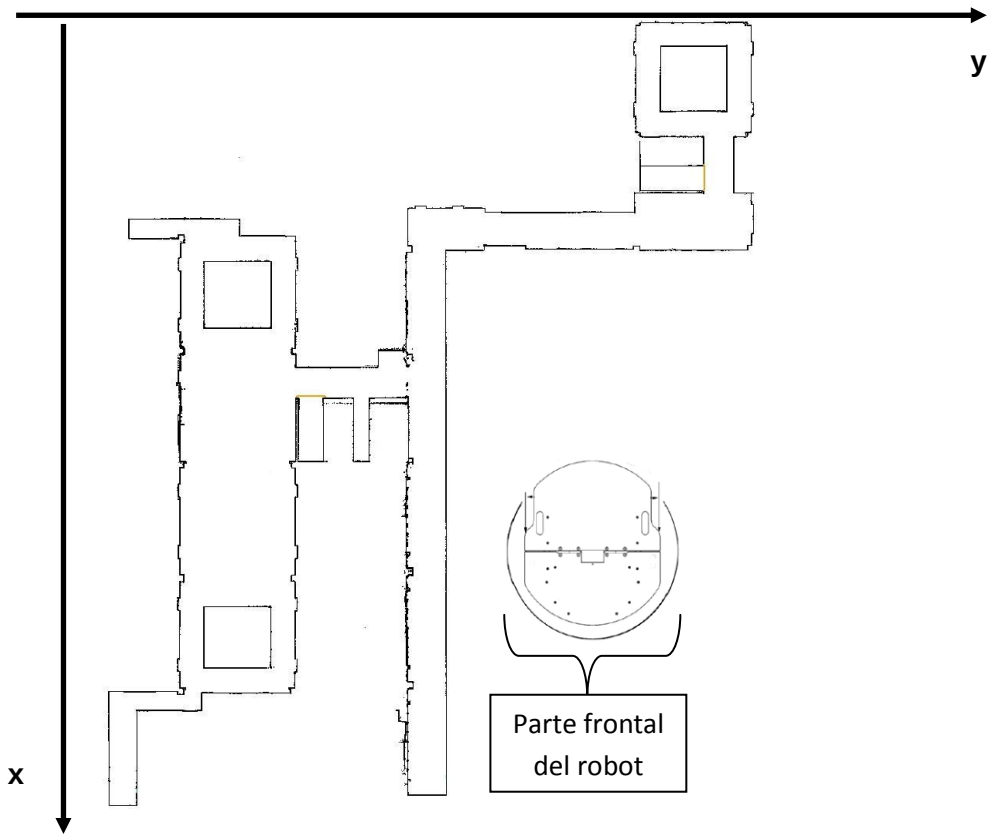



Figura 49. Sistema de coordenadas de la implementación del Sistema de Navegación Topológica

### 4.1.3 Módulo Mapa Topológico

En el Mapa Topológico se almacenan los nodos, donde cada nodo representa cada una de las marcas que el robot debe reconocer durante su navegación. Cada nodo está puesto de tal forma que haya una relación espacial entre ellos donde se pueda unir uno con otro por medio de un segmento rectilíneo y que no intercepte ningún obstáculo. La finalidad de este es poder localizar al robot dentro del entrono de navegación.

En la Figura 50 se muestra el mapa topológico construido del segundo piso de la escuela de Ingeniería Electrica y Electrónica.

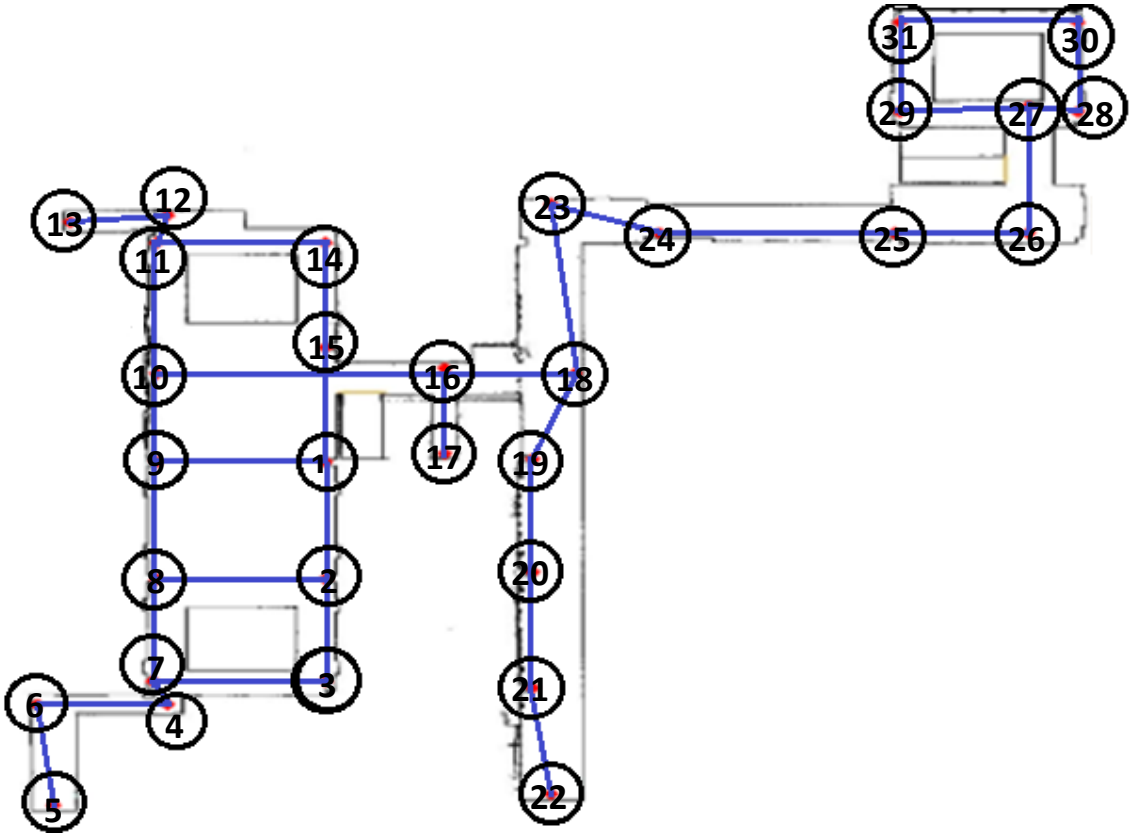


Figura 50. Mapa topológico construido del segundo piso de la escuela de Ingeniería Electrica y Electrónica.

### 4.1.4 Módulo Captura

Para implementar el modulo de captura se utiliza de la librería OpenCV la función *cvCaptureFromCAM(int index),* que localiza e inicializa la estructura CVCapture para leer la secuencia de imágenes de la cámara de video, actualmente esta función puede utilizar dos interfaces en Linux, V4L y FireWire. El parámetro *index* de la función es el índice a la cámara que se utiliza dentro del sistema operativo.
La imagen adquirida es capturada por la función cvQueryFrame(CvCapture* capture), descomprimida y devuelta en una estructura *IpImage* para ser visualizada y manipula posteriormente. En caso de ocurrir un error en la captura de la imagen se arroja un mensaje de error y el programa se detendrá. Antes de finalizar el programa se realiza la liberación del controlador del dispositivo por medio de la función *cvReleaseCapture(&capture).*

### 4.1.5 Módulo Corrección

Una vez capturadas las imágenes se procede a calcular los parámetros intrínsecos y extrínsecos de la cámara que junto con la distorsión son los elementos del modelo de calibración de la cámara utilizado con OpenCV. Las funciones de OpenCV implementadas utilizan el modelo de cámara pinhole **(ver Capitulo 2 sección 2.7.3.1)**

La calibración de la cámara se hace de acuerdo al modelo descrito anteriormente utilizando una serie de imágenes tomadas a una grafica impresa similar a un tablero de ajedrez de 10x7 cuadrados de lado 3 cm, la cual se utiliza como patrón (Figura 51).

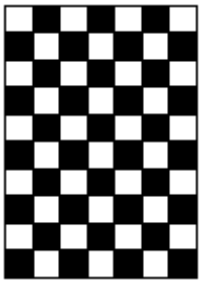


Figura 51. Patrón utilizado para la calibración y corrección de la distorsión de la imagen.

Por medio de la función *cvFindChessBoardCorners()* se encuentra la posición de las esquinas internas de los cuadros como se muestra en la Figura 52, y los puntos encontrados son almacenados en la variable *corners*. La posición de las esquinas es empleada por la función *cvCalibrateCamera()*, esta obtiene los parámetros intrínsecos de la cámara usando los puntos del patrón objeto y el patrón imagen, y almacena la matriz de parámetros en la variable *camera_matrix.*

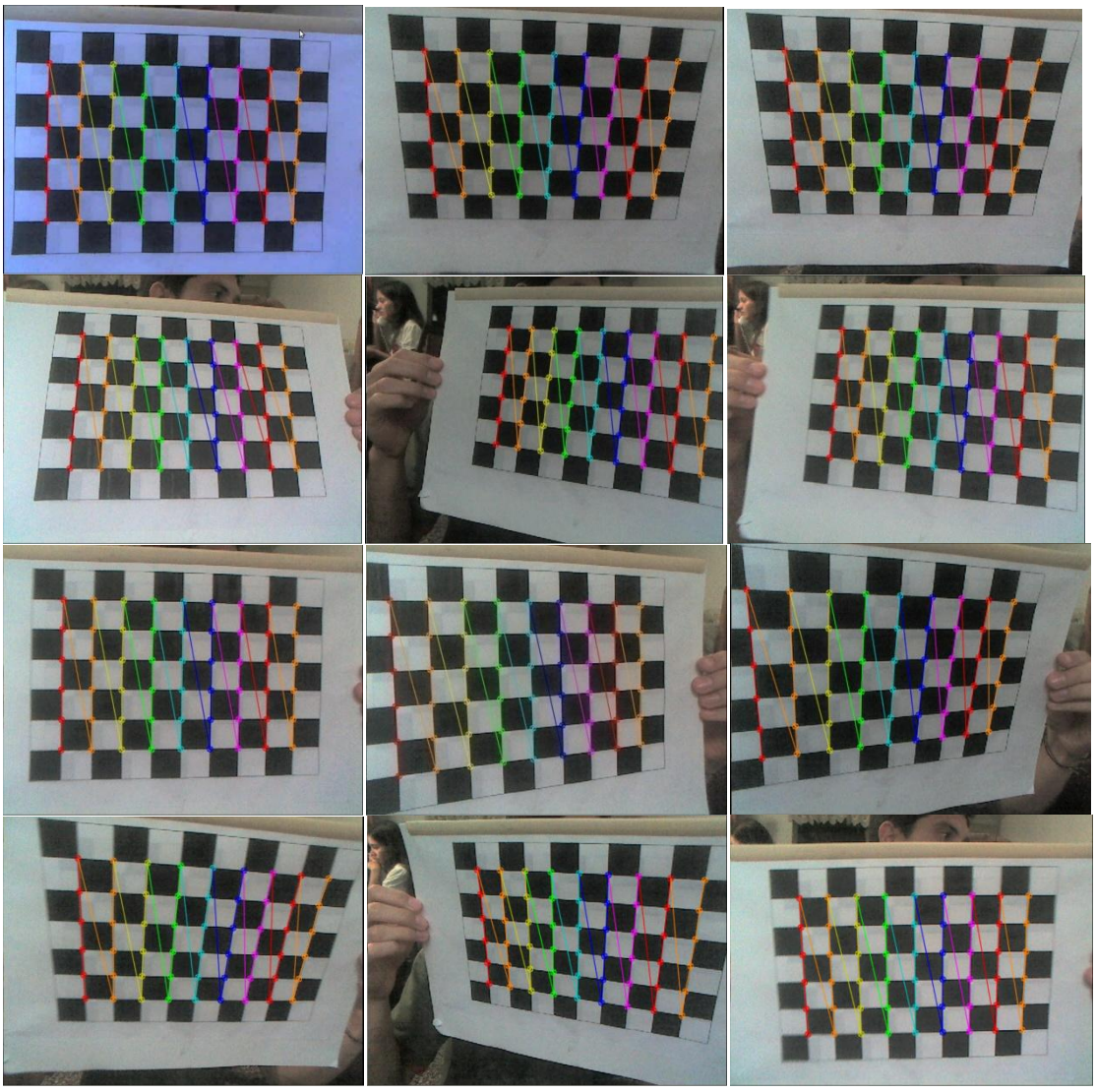


Figura 52. Imágenes utilizadas para la calibración de la cámara.

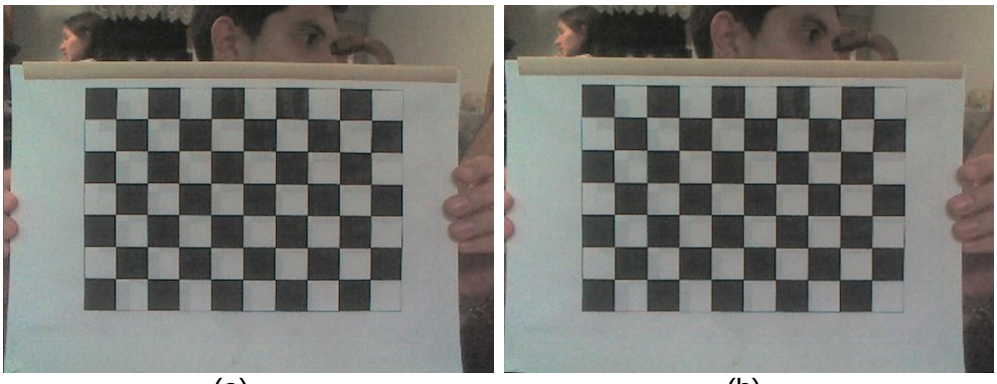
Los parámetros extrínsecos son encontrados utilizando la función *cvFindExtrinsicParams()* en donde se utiliza la longitud focal y el punto principal hallados en el paso anterior. Las variables *rotation_vector* y *translation_vector* contienen los parámetros resultantes.

Luego se procede a corregir la distorsión de la imagen capturada por la cámara por medio de la función *cvUnDistortOnce(const CvArr* src, CVArr* dst, const float* intrinsic_matrix, const float* distortion_coeffs, int interpolate=1)* la cual genera una nueva imagen corregida almacenándola en la variable *dst*.

A continuación se describen cada uno de los parámetros de la función ***cvUnDistortOnc:***

- ***src*** es la variable donde se carga la imagen distorsionada.
- ***dst*** es la variable donde se almacena la nueva imagen corregida.
- ***intrinsic_matrix*** es una matriz 3x3 donde se almacenan los parámetros intrínsecos de la cámara.
- ***distortion_coeffs*** es donde se almacenan los cuatro coeficientes de distorsión fx, fy, cx y cy.
- ***Interpolate*** es la bandera de interpolación bilinear

En la figura 53 se puede apreciar el resultado de la corrección.



(a) (b)

Figura 53. (a) Imagen distorsionada. (b) Resultado de la corrección de la distorsión

### 4.1.6 Módulo Conversión a escala de grises

La imagen corregida se pasa a escala de grises por medio de la función *cvCvtColor( g_image, g_gray, CV_BGR2GRAY )* de la librería OpenCV, donde *g_image* es la

variable donde se carga la imagen fuente, *g_gray* es donde se almacena la nueva imagen la cual se desea pasar a escala de grises y *CV_BGR2GRAY* es la operación de conversión de modelo color de BGR a escala de grises.

### 4.1.7 Módulo Mejoramiento

Se le aplica un filtro mediana a la imagen para eliminar ruido del tipo *salt&peper* utilizando tamaño de ventanas de 3 pixeles de lado ya que de acuerdo a la bibliografía consultada con este tamaño se obtienen mejores resultados. Para la implementación de este filtro se utiliza la función *cvSmooth(const CvArr* src, CvArr* dst, int smoothtype = CV_MEDIAN, 3, 3),* donde *src* es la variable donde se guarda la imagen fuente*, dst* es la variable donde se almacena la nueva imagen con el filtro aplicado *y smoothtype* es el tipo de filtro de suavizado que se desea aplicar a la imagen fuente*.* Posteriormente se realiza una operación morfológica *closing* que consiste en una erosión seguida de una dilatación por medio de las funciones *cvErode(IplImage* src, IplImage* dst, IplConvKernel* B = NULL, int iterations = 1)* y *cvDilate(IplImage* src, IplImage* dst, IplConvKernel* B = NULL, int iterations = 1)* respectivamente.

### 4.1.8 Módulo Reconocimiento de marcas

Inicialmente se propuso construir figuras de formas geométricas con relleno de color negro sobre un papel blanco como se muestra en la Figura 54.

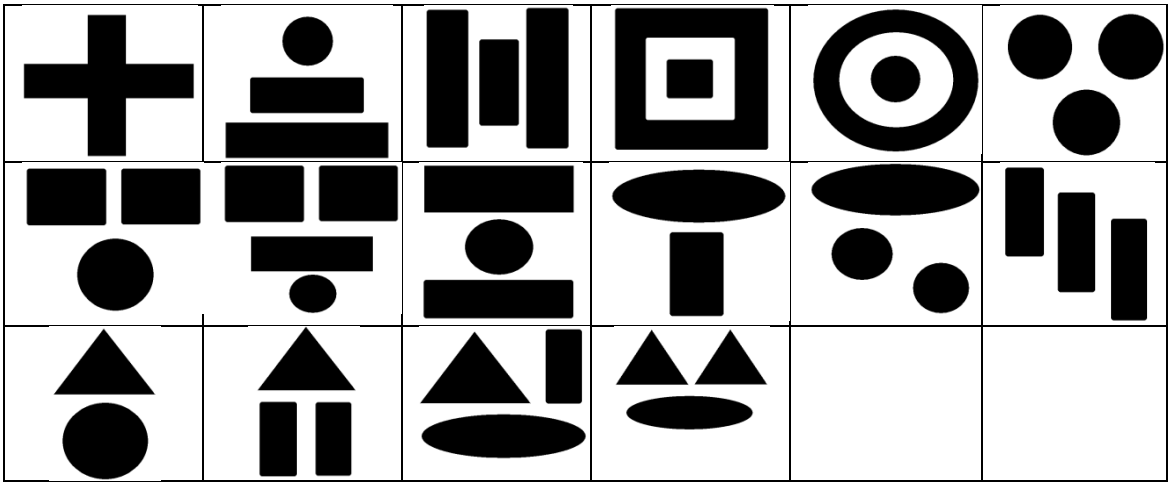


Figura 54. Tipo de marcas que se pensaron utilizar en un principio.

A la hora de planear la trayectoria para llevar a cabo la navegación del robot se encontró que este tipo de marcas no eran las más adecuadas debido que el robot por medio de su sistema de visión no siempre se las va a encontrar de frente y la marca pasaría desapercibida lo que provocaría que el robot se perdiera en su trayectoria.
Posteriormente se trató de darle una solución a este problema el cual consistió en aprovechar que la cámara puede girar horizontalmente y en caso que la marca no estuviera de frente al robot, al hacerla girar la podría reconocer, pero esto también tiene una desventaja, ya que en el momento en que el robot se desplace cerca a la marca y la cámara esté orientada en otra dirección, ésta también pasaría desapercibida.

Revisando la bibliografía se optó por utilizar marcas inspiradas en las usadas en el Concurso de Robots Móviles AAAI **[DEAN 93]** donde cada marca son códigos de

barras cilíndricas de tal manera que siempre van  a ser percibidas de forma igual sin importar el punto de observación. Para la construcción de estas se emplearon tubos de ventilación de 35 cm de altura cada uno y de un diámetro de 4.8 cm lo que permite ubicarlos en posición vertical, a los cuales se les diseñó un código de barras de 6 franjas de igual ancho de colores blanco y negro. El color de las franjas de cada marca se intercaló de tal forma que se pudiera diferenciar una marca de otra y así construir un total de 31 marcas, las cuales se distribuyeron en el mapa topológico construido del segundo piso de la Escuela de Ingeniería Eléctrica y Electrónica.

Sin embargo el reconocimiento de este tipo de marcas no fue satisfactorio ya que debido a que las franjas son de color blanco y negro; y las paredes del entorno son blancas y las columnas son grises, a la hora del reconocimiento de landmarks el color de las paredes y de las columnas se confundía como si fuera una franja más y se reconocían como marcas aquellas que no se esperaban. Para esto se eligió utilizar parte del tubo de ventilación aprovechando que todos son amarillos y colocar una franja más pero de este color y así se evitaría el inconveniente mencionado. Por tal motivo no se siguió trabajando en escalas de grises sino con imágenes a color.

En la figura 55 se muestra una foto de una de las marcas que se construyeron con un tubo de ventilación.

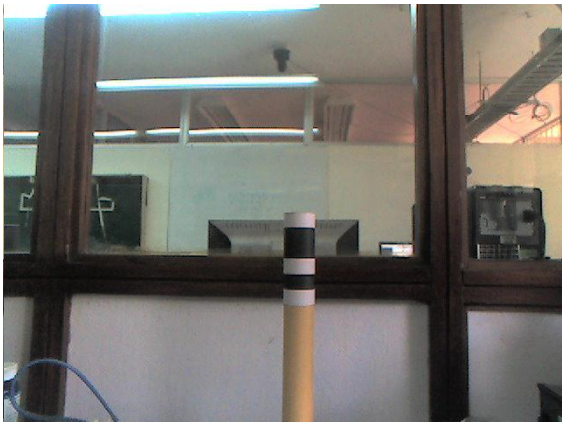


Figura 55. Fotografía de una de las marcas que se construyeron con un tubo de ventilación

En la Tabla 3 se muestra la base de datos de las 31 marcas que se construyeron, a las cuales se les asignó un número diferente.

Tabla 3. Base de datos de las 31 marcas construidas con tubo de ventilación

| | MARCAS | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| | MARCAS | | | | | | | | | | | | | | | | |
| # | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | | | |

Luego de hacer unas pruebas de reconocimiento el sistema de reconocimiento siguió presentando la misma falla, por lo que se optó por usar un color que resaltara bastante y fuera percibido para entornos con buena iluminación como para entornos con poca luz. Este color es el magenta el cual es un muy visible para ambos casos. Las marcas se dejaron con las mismas franjas de color blanco y negro y se les añadió una franja de color magenta en la parte superior e inferior a cada tubo. Al realizar algunas pruebas de reconocimiento se observó que se logró resolver los problemas mencionados anteriormente.

En la Tabla 4 se muestra la base de datos de las 31 marcas utilizadas en la aplicación, las cuales se distribuyeron en el segundo piso de la escuela de Ingeniería Eléctrica y Electrónica.

Tabla 4. Base de datos de las 31 marcas construidas con tubo de ventilación utilizadas en la aplicación.

| | MARCAS | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| | MARCAS | | | | | | | | | | | | | | | | |
| # | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | | | |

Para el reconocimiento de marcas se emplea el algoritmo ***match template*** por medio de la función *cvMatchTemplate( const CvArr* image, const CvArr* templ, CvArr* result, int method )* donde el patrón a buscar (marca) se compara con la imagen adquirida por la cámara para encontrar el área más similar en ésta. El patrón se va pasando contra todas las posibles posiciones de ella y se calcula un valor de función de similitud (o distancia) en cada posición. El resultado de las comparaciones se va almacenando en una matriz adicional. La posición donde alcance el máximo de la función de similitud (o, mínimo de una función de distancia), será aquella donde se puede decir que se encuentra el patrón. La variable *image* es donde se carga la imagen adquirida por la cámara y a la que se le ha hecho su debido tratamiento, *templ* es la variable donde se guarda el patrón a buscar, *result* es la variable donde se almacena la matriz del resultado de las comparaciones y *method* es el método que se va a usar para comparar el patrón con las regiones de la imagen. Estos métodos son:

- CV_TM_SQDIFF (diferencia al cuadrado)
- CV_TM_SQDIFF_NORMED (cuadrado de la diferencia normalizada)
- CV_TM_CCORR (correlación cruzada)
- CV_TM_CCORR_NORMED (correlación cruzada normalizada)
- CV_TM_CCOEFF (coeficiente de correlación)
- CV_TM_CCOEFF_NORMED (coeficiente de correlación normalizada)

Después de probar con cada uno de los métodos mencionados se observó que se obtuvieron mejores resultados con el método *CV_TM_CCOEFF_NORMED* (Correlación normalizada) donde el valor de la similitud se encontrará en la posición del máximo. Para buscar los máximos y mínimos se usa la función *cvMinMaxLoc (const CvArr* A, double* minVal, double* maxVal, CvPoint* minLoc, CvPoint* maxLoc, const CvArr* mask=0)*, donde minVal y maxVal son el mínimo y el máximo global en A, respectivamente; y minLoc y maxLoc son la posición (x, y) del mínimo y máximo en A. Posteriormente se dibuja un contorno rectangular con bordes rojos en la imagen ya procesada donde está localizado el patrón en esta por medio de la función *cvRectangle(frame2, maxloc, cvPoint(maxloc.x+tpl->width, maxloc.y+tpl->height), CV_RGB(255,0,0), 3); cvRectangle(res, cvPoint(maxloc.x-tpl->width, maxloc.y-tpl->height), cvPoint( maxloc.x + tpl->width, maxloc.y+ tpl->height), cvScalarAll(0), -1); cvMinMaxLoc(res, &minval, &maxval, &minloc, &maxloc)*.

### 4.1.9 Módulo Planeador Misión

Tiene la responsabilidad de la planificación de alto nivel, de la cual se encarga el usuario al establecer los puntos de inicio y destino para llevar a cabo la navegación.

### 4.1.10 Módulo de Navegación

Genera la trayectoria adecuada que debe seguir el Robot móvil para alcanzar la marca destino desde la posición inicial o actual en la que se encuentre éste. La planeación de la trayectoria está basada en el algoritmo por grafos de visibilidad que corresponde a los arcos que unen cada nodo del Mapa topológico, en efecto, esos nodos tiene una relación espacial y cada uno representa una marca de referencia que debe reconocer el Robot para alcanzar su objetivo. Para que el robot navegue de un nodo a otro se utiliza el algoritmo VFH el cual utiliza un descriptor basado en el algoritmo de campos de potencial, donde de acuerdo en la marca de referencia en que se encuentre y la relación que tiene con la siguiente a alcanzar este modulo fijará la dirección en la que el robot debe navegar para encontrar la siguiente y en el momento en que se presente un obstáculo el robot lo evadirá y seguirá dirigiéndose hacia la marca. Para llevar a cabo el algoritmo VFH es necesario dar una consigna y se usa la de 10 m en la dirección de la siguiente marca. Se da una distancia de 10 m debido a que el mapa no es métrico, por lo tanto no se conocen las distancias entre cada punto de referencia, pero si se sabe que entre marca y marca las distancias son inferiores a 10 m. El robot recorrerá 10 m en dirección hacia dicha marca hasta que la encuentre, la cual una vez reconocida, se dará al robot otro punto a 10 m en dirección de la siguiente marca a alcanzar y así sucesivamente hasta llegar a la marca destino.

Constantemente este módulo está en comunicación con el del Planeador Misión, por medio del Mapa Topológico, donde se mantiene información acerca de las marcas visibles, reportándole el estado de la navegación, como su posición a través de su localización en dicho Mapa y reportándole el reconocimiento de cada una de las marcas a través de su sistema de visión. Las distancias a los obstáculos que encuentra en su camino las reporta por medio del módulo Player a través de la interface sonar. Estos datos están disponibles para ser usados por el Módulo Piloto en la selección y ejecución de comportamientos.

### 4.1.11 Módulo Piloto

Se encarga de ejecutar las trayectorias generadas nodo a nodo por el Módulo Navegador seleccionando los comportamientos apropiados proporcionados por el

Módulo Comportamientos permitiéndole al Robot un desplazamiento seguro. Estos comportamientos los parametriza este módulo y se los envía al Módulo Player.
En caso tal que el robot en su trayectoria pase por alto una marca, éste seguirá navegando hacia el punto dado por el módulo navegador. Si en el transcurso de la trayectoria recorrida se reconoce uno de los puntos de referencia del entorno, el módulo piloto se encargará de reportar al módulo planeador de misión: que no se cumplió con la submeta establecida y a partir de la marca actual en la que se encuentra, el módulo planeador de misión generará la nueva secuencia de marcas que se deben reconocer hacia la marca destino y el módulo navegador se encargará de generar una nueva trayectoria en base a esta nueva secuencia de marcas, la cual será ejecutada por el módulo piloto.

### 4.1.12 Módulo Comportamientos

Se encarga de administrar la dirección de movimiento que tiene asociado cada marca de referencia del entorno, los comandos de velocidad y rotación que se darán a los actuadores del robot. Los comportamientos que se le dan al robot son los siguientes:

- ***Moverse hacia un punto*.** Para realizar este comportamiento se utiliza la función *playerc_position2d_set_cmd_pos_with_vel* explicada en el Módulo Player.
- ***Evadir obstáculo.*** Para la implementación de este comportamiento se utiliza el algoritmo VFH.
- ***Encontrar un punto de referencia.*** Este comportamiento se lleva a cabo por medio del Módulo Reconocimiento.
- ***Parar.*** Para realizar este comportamiento se utiliza la función *playerc_position2d_set_cmd_pos_with_vel* explicada en el Módulo Player.

### 4.1.13 Módulo Localización

Para la localización se emplean las marcas de referencia donde si el robot encuentra una de estas en el entorno y esa aparece en el mapa topológico, entonces el robot se localiza en relación con este. Si el módulo navegador planea una ruta, los puntos de referencia sirven para advertirle al módulo piloto cuando se ha completado un segmento y cuando otro debe empezar.

### 4.1.14 Módulo Sonares

Se encarga de calcular la distancia promedio de los obstáculos en el entorno próximo al Robot y transfiere la información al procesador principal de éste. Estos datos son obtenidos por el programa cliente por medio de la Interface “sonar:0” y el driver “p2os” proporcionados por la herramienta software Player.

### 4.1.15 Módulo Odometría

Se encarga de calcular la posición local del Robot, esta localización se da en términos de coordenadas, desde un punto de partida de la Plataforma Móvil (Posición (0; 0) a una orientación 90°) hasta el punto donde se encuentre ubicado con respecto a la posición inicial. Para esto se utiliza el driver “p2os” de la herramienta software Player el cual provee de una interfaz *position2d* que permitir mandar comandos de velocidad y/o ubicación de éste y recibir la información en dos coordenadas.

### 4.1.16 Módulo Batería

Proporciona al usuario la carga de la batería del Robot para en caso de estar agotada, poner a recargarla. Este dato es obtenido por el programa cliente por medio de la Interface “power:0” y el driver “p2os” proporcionados por la herramienta software Player.

## 4.2 CONCLUSIONES

Se construyó un mapa topológico del entorno de navegación (segundo piso de la escuela de Ingeniería Eléctrica y Electrónica) distribuyendo las 31 marcas de referencia construidas a través de todo el entorno formando un grafo o red de nodos y aristas, donde cada nodo representa una marca y las aristas los nodos navegables entre ellas. Cada marca artificial tiene forma cilíndrica y se construyeron de forma artesanal con un tubo de ventilación, ya que la detección de estas por medio de una cámara Web es mucho más sencilla, al ser conocido su tamaño y forma, permitiendo poder determinar la posición del robot.

Se implementó un sistema de navegación topológica relacional que le permite al robot desplazarse desde una marca inicial hasta una final evadiendo los obstáculos, para lo cual se desarrolló un sistema de visión para la detección de marcas artificiales del entorno. Para la navegación del robot se tiene un conocimiento a priori de cómo ir de una marca a otra y se utiliza el algoritmo VFH, el cual utiliza un descriptor basado en el algoritmo de campos de potencial apoyado en técnicas reactivas.

La implementación de una arquitectura de control híbrida reactiva/deliberativa presenta varias ventajas ya que se distribuye la complejidad de la tarea de navegación en 2 niveles: en el nivel inferior la capa reactiva que se ejecuta en el robot y se encarga de la evasión de obstáculos en el entorno local y alcanzar metas puestas por el nivel deliberativo; en el nivel superior la capa deliberativa se encarga de crear la ruta para ir de un punto a otro, llevar a cabo la navegación entre estos y relocalizar al robot dentro del mapa topológico. Otra ventaja de este tipo de arquitectura es que tiene un diseño modular, lo que permite dividir el sistema en subsistemas menores (conocidos como módulos) muy bien diferenciados, lo que proporciona que unos módulos puedan ser reemplazados por otros sin problemas, lo que permite una flexibilidad muy grande en la implementación del sistema.

El subsistema de visión ofrece la realimentación del estado del sistema al subsistema deliberativo y así poder llevar a cabo de manera satisfactoria la navegación entre marcas, al identificar y reconocer las 31 marcas artificiales distribuidas en el entorno.

El uso de las librerías de OpenCV para el procesamiento de imágenes presenta varias ventajas ya que ofrece funciones de alto nivel tales como: captura, calibración, erosión, dilatación y visualización. Además de ello cuenta con funciones para liberar la cabecera y los datos de las imágenes lo que permite un mayor rendimiento computacional.

Para el reconocimiento de imágenes se implementó el algoritmo template matching (comparación de plantillas) con el uso de las librerías de OpenCV con lo que se logró hasta un reconocimiento satisfactorio del 86%, que depende de tres factores: iluminación del entorno, distancia entre la cámara y la marca; y del ángulo con que se capture la marca.

Se realizó un debido proceso de calibración de la cámara que le permite al sistema de visión obtener y calcular parámetros para corregir la distorsión de perspectiva. Este proceso junto con el método de reconocimiento de patrones implementado, facilitó y se obtuvo un eficaz reconocimiento de las marcas a partir de las imágenes capturadas.

Se desarrolló una interfaz gráfica de usuario (GUI) para facilitar la interacción de éste con el sistema de navegación topológica implementado. Por medio de esta el usuario puede conectarse vía Wi-Fi a la plataforma móvil Pioneer 3DX en la cual corre el programa servidor; visualizar el entorno por donde navega el Robot, así como el mapa topológico construido del segundo piso de la Escuela de Ingeniería Eléctrica y Electrónica con la ubicación cada uno de los puntos de referencia. La interfaz también posee dos paneles que permiten indicarle al robot la marca inicial en que se encuentra y la marca destino a la cual se desea que navegue el robot. Una vez el usuario ingrese ambas marcas, se mostrará en el mismo mapa topológico la ruta a seguir desde la marca donde se encuentre el Robot hasta la marca destino; como también la posición en la que se encuentre el Robot, una vez reconozca la marca. Aparte de esta GUI se utiliza una de las interfaces proporcionados por el software Player, en la cual se muestra gráficamente la lectura del anillo de sonares del robot; y la odometría y la velocidad del robot en las coordenadas $x, y, \theta$.

El uso de software libre como Player y OpenCV facilitó implementar los algoritmos necesarios explicados en cada uno de los módulos del sistema de visión, así como todos los comportamientos proporcionados al robot, y la comunicación del programa cliente con el servidor.

# CAPITULO V

## 5. PRUEBAS Y RESULTADOS

En este capítulo se presenta la descripción de las pruebas que se efectuaron al sistema de navegación topológica para validar su funcionamiento y se consignan los resultados y el análisis de los mismos, con lo cual se permite definir características, alcances y limitaciones del sistema implementado.

Las pruebas se realizan en el segundo piso de la escuela de Eléctrica y Electrónica, el cual es un entorno estructurado y apto para llevar a cabo la validación del sistema de navegación implementado.

Para controlar el Robot Pioneer 3DX, este se conecta vía Wifi al Router Wi-Fi LynkSys WRT54G a la red inalámbrica “Pioneer3DX” en la dirección 192.168.1.101. El PC HP Pavilon G4, con sistema operativo Ubuntu 10.04, donde corre el programa cliente se conecta a esta misma red con el fin de poder conectarse posteriormente con el servidor Player que se está ejecutando en el robot.

Gracias a las pruebas realizadas se pueden sacar ciertas conclusiones y realizar valoraciones que posteriormente pueden ser fundamentales para el éxito del proyecto y de los futuros trabajos apoyados en él.

### 5.1 PRUEBAS PARA EL SISTEMA DE PROCESAMIENTO DIGITAL DE IMÁGENES Y RECONOCIMIENTO DE MARCAS

#### 5.1.1 Prueba para caracterizar el sistema de visión

El objetivo de esta prueba es calcular el ángulo de visión de la cámara y a que distancias las marcas pueden ser visualizadas por esta. Para esto se colocó la cámara sobre el robot Pioneer 3DX y a partir del lente de la cámara se tomaron 6 ángulos con respecto a la horizontal obteniendo ángulos cada 30° como se muestra en la Figura 56

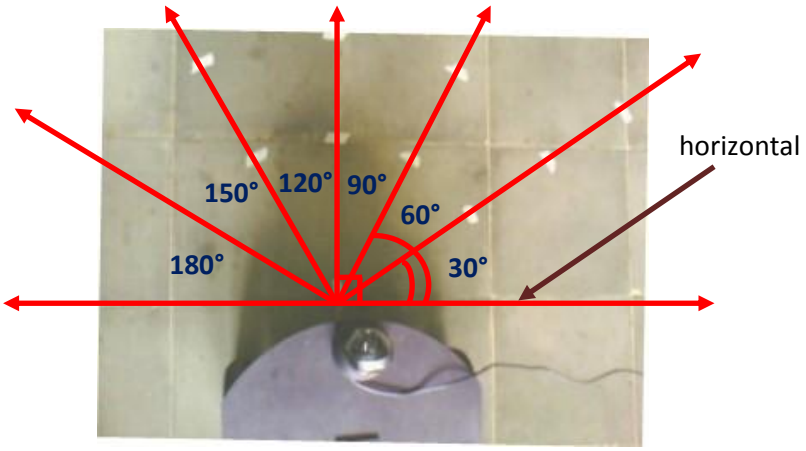



Figura 56. Fotografía de la prueba para el cálculo del ángulo de visión de la cámara.

Luego se colocaron las marcas a una distancia de 1.05 m de la cámara (distancia a la cual la marca puede ser totalmente capturada), como se muestra en la Figura 57, para cada uno de los ángulos.

Figura 57. Fotografía de una marca a 1.05 m de la cámara para calcular el ángulo de visión del sensor de visión.

A continuación se muestra la imagen capturada por la cámara para ángulos de 60°, 90° y 120° con respecto a la horizontal.


(a) (b) (c)

Figura 58. Imagen capturada de la marca a 1.05 m para un ángulo de: (a) 60° (b) 90° (c) 120°, con respecto a la horizontal.

De acuerdo a la Figura 58 se puede observar que para ángulos de 0° a 60° y de 120° a 180° las marcas se saldrían del ángulo de visión de la cámara, por lo tanto se pude concluir que este se encuentra entre 60° y 120° con respecto a la horizontal. En base a esto se concluye que la cámara posee un campo de visión de 60°.

Una vez obtenido el ángulo de visión de la cámara se procede a calcular la distancia a partir de la cual se pueden observar las marcas. Para esto se observó la imagen capturada, por medio de la GUI y la marca se fue moviendo poco a poco hasta que esta se pudiera observar. La distancia mínima a la que la marca se pudo observar fue a 30 cm, como se observa en la Figura 59.


(a) (b) (c)

Figura 59. Imagen capturada de la marca por la cámara a una distancia de 30 cm para una ángulo de (a) 60° (b) 90° (c) 120°, con respecto a la horizontal.

### 5.1.2 Pruebas para el reconocimiento de marcas

El objetivo de estas pruebas es calcular la distancia a la cual las 31 marcas son reconocidas por medio de la cámara Web dentro del ángulo de visión de esta, para diferentes lugares con diferente tipo de iluminación.

#### *5.1.2.1 Lugar donde la luz incide directamente sobre las marcas*

Esta prueba se realizo en un lugar del espacio de Robotica Móvil donde la luz solar incide directamente sobre la marca capturada por el sistema de visión como se muestra en la Figura 60. Estas pruebas se realizaron entre las 10 de la mañana y 5 de la tarde, en un día soleado.

Figura 60. Fotografías donde la luz del sol incide directamente sobre las marcas.

En la tabla 5 se muestra el registro del reconocimiento de cada una de las 31 marcas en este lugar, para un ángulo de 60°, 90° y 120° con respecto a la horizontal, a diferentes distancias. Los "✓" representan que la marca fue reconocida y las "x" lo contrario.

Tabla 5. Registro del reconocimiento de marcas para un lugar donde la luz solar incide directamente sobre la marca.

| Dist | 30 cm | | | 50 cm | | | 70 cm | | | 80 cm | | | 85 cm | | | 90 cm | | | 95 cm | | | 1 m | | | 1.05 m | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Marca / Ángulo | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° |
| 1 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 3 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 4 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 5 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 8 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 9 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 10 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 11 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 12 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 13 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 14 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 15 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 16 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 17 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 18 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 19 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 20 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 21 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 22 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 23 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 24 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 25 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 26 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 27 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 28 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 29 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 30 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 31 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PMR | 0% | | | 0% | | | 0% | | | 0% | | | 0% | | | 0% | | | 0% | | | 0% | | | 0% | | |

**PMR: Promedio Marcas Reconocidas**

Con respecto a los resultados obtenidos en la tabla 5, se concluye que para un lugar donde la luz solar incide directamente sobre las marcas y con la cámara enfrente de estas, ninguna de las marcas es reconocida.

### *5.1.2.2 Lugar donde la iluminación es controlada*

Esta prueba se realizo en un lugar del espacio de Robótica Móvil donde la iluminación es controlada, donde la mayor parte de la iluminación depende de lámparas fluorescentes como se muestra en la Figura 61.


Figura 61. Fotografías del lugar del espacio de Robótica Móvil con iluminación controlada.

En la tabla 6 se muestra el registro del reconocimiento de cada una de las 31 marcas en este lugar, para un ángulo de 60°, 90° y 120° con respecto a la horizontal, a diferentes distancias. Los "✓" representan que la marca fue reconocida y las "x" lo contrario.

Tabla 6. Registro del reconocimiento de marcas para un lugar con iluminación controlada.

| Dist | 30 cm | | | 50 cm | | | 70 cm | | | 80 cm | | | 85 cm | | | 90 cm | | | 95 cm | | | 1 m | | | 1.05 m | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Marca / Ángulo | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° |
| 1 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 2 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 3 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 4 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 5 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 6 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 7 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 8 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 9 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 10 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 11 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 12 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 13 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 14 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 15 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 16 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 17 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 18 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 19 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 20 | x | x | x | x | x | x | x | x | x | ✓ | ✓ | ✓ | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 21 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 22 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 23 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 24 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 25 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 26 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 27 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 28 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 29 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 30 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 31 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3.2 | 100 | 3.2 | 16.1 | 100 | 16.1 | 100 | 100 | 100 | 100 | 29 | 100 | 100 | 0 | 100 | 0 | 0 | 0 |
| PMR | 0% | | | 0% | | | 0% | | | 35.5% | | | 44.1% | | | 100% | | | 76.3% | | | 66.7% | | | 0% | | | |

**PMR: Promedio Marcas Reconocidas**

De acuerdo a los resultados obtenidos en la tabla 6, se puede determinar que para un ángulo de 60° y 120° con respecto a la horizontal, todas las 31 marcas se reconocen entre 90 cm y 1 m a distancia de la cámara. Por otro lado si la marca se encuentra en frente de la cámara, es decir, a 90° con respecto a la horizontal, todas las 31 marcas son reconocidas a una distancia entre 80 y 90 cm.

#### *5.1.2.3 Lugar donde la iluminación es variable*

Esta prueba se realizo en el pasillo del edificio 353 donde la iluminación es variable, ya que ésta depende de las condiciones ambientales de la luz como se muestra en la Figura 62.

Figura 62. Fotografías del pacillo del edificio 353 con iluminación variable.

En la tabla 7 se muestra el registro del reconocimiento de cada una de las 31 marcas en este lugar, para un ángulo de 60°, 90° y 120° con respecto a la horizontal, a diferentes distancias. Los "✓" representan que la marca fue reconocida y las "x" lo contrario.

Tabla 7. Registro del reconocimiento de marcas para un lugar con iluminación variable.

| Dist | 30 cm | | | 50 cm | | | 70 cm | | | 80 cm | | | 85 cm | | | 90 cm | | | 95 cm | | | 1 m | | | 1.05 m | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Marca / Ángulo | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° | 60° | 90° | 120° |
| 1 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 2 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 3 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | ✓ | x | x | x |
| 4 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 5 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 6 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 7 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 8 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 9 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 10 | x | x | x | x | x | x | x | x | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 11 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | ✓ | x | ✓ | x | x | x |
| 12 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 13 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 14 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 15 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 16 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | ✓ | x | ✓ | x | x | x |
| 17 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 18 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 19 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | ✓ | x | ✓ | x | x | x |
| 20 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | x | ✓ | x | x | x |
| 21 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x |
| 22 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 23 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 24 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | ✓ | x | ✓ | x | x | x |
| 25 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | ✓ | x | ✓ | x | x | x |
| 26 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | x | x | x | ✓ | x | ✓ | x | x | x |
| 27 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 28 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | x | ✓ | x | x | x | x | x | x | x | x | x | x |
| 29 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 30 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| 31 | x | x | x | x | x | x | x | x | x | x | ✓ | x | x | ✓ | x | ✓ | ✓ | ✓ | ✓ | x | ✓ | ✓ | x | ✓ | x | x | x |
| % | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 6.5 | 100 | 6.5 | 58.1 | 100 | 58.1 | 54.8 | 32.3 | 51.6 | 74.2 | 0 | 74.2 | 0 | 0 | 0 |
| PMR | 0% | | | 0% | | | 0% | | | 33.3% | | | 37.7% | | | 72.1% | | | 46.2% | | | 49.5% | | | 0% | | |

**PMR: Promedio Marcas Reconocidas**

En base a los resultados obtenidos en la tabla 6, se puede determinar que para un ángulo de 60° y 120° con respecto a la horizontal, no todas las marcas son reconocidas para un lugar con iluminación variable. Pero para un ángulo de 90° con

respecto a la horizontal, todas las 31 marcas son reconocidas a una distancia entre 80 y 90 cm.

## 5.2 PRUEBAS PARA EL SISTEMA DE NAVEGACIÓN TOPOLÓGICA

En cada una de las pruebas, el robot realiza una navegación desde una marca inicial a una final, las cuales son indicadas desde la interfaz gráfica de usuario, y llevando a cabo trayectorias diferentes: en línea recta, en forma de “L” y en forma de “U”.

Todas las misiones (pruebas) se desarrollaron sobre el entorno mostrado en la Figura 41, cuyo mapa topológico se muestra en la Figura 50.

### 5.2.1 Trayectoria en línea recta

En esta prueba se procede a realizar dos misiones donde la trayectoria recorrida es en línea recta. La primera misión realizada es “a partir del punto de referencia 1 dirigirse al punto destino 2”, la cual se realizó nueves veces; la segunda misión realizada es “a partir del punto de referencia 26 dirigirse al punto destino 23” la cual se realizó 5 veces. Estas pruebas se realizaron con el fin de observar si el odómetro es capaz de representar el movimiento recto fielmente, o si por el contrario comete un error en la medición y en ése caso poder cuantificarlo.

En la Figura 63 se muestra por medio de punto rojos la ubicación de cada una las marcas del entorno; las líneas azules representan la ruta generada por el módulo navegador para desplazarse a la marca 2 desde la marca 1; y el punto verde indica la ubicación en que se encuentra el robot cada vez que ha reconocido alguna de las marcas por medio de su sistema de visión.

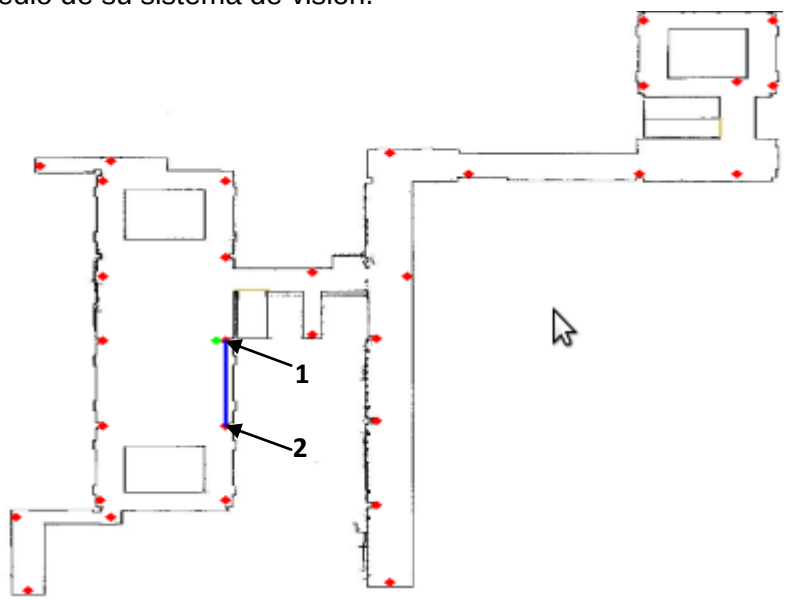



Figura 63. Mapa Topológico con la trayectoria generada por el módulo de navegación para desplazarse de la marca 1 a la 2.

A continuación se muestra la trayectoria descrita por el robot:

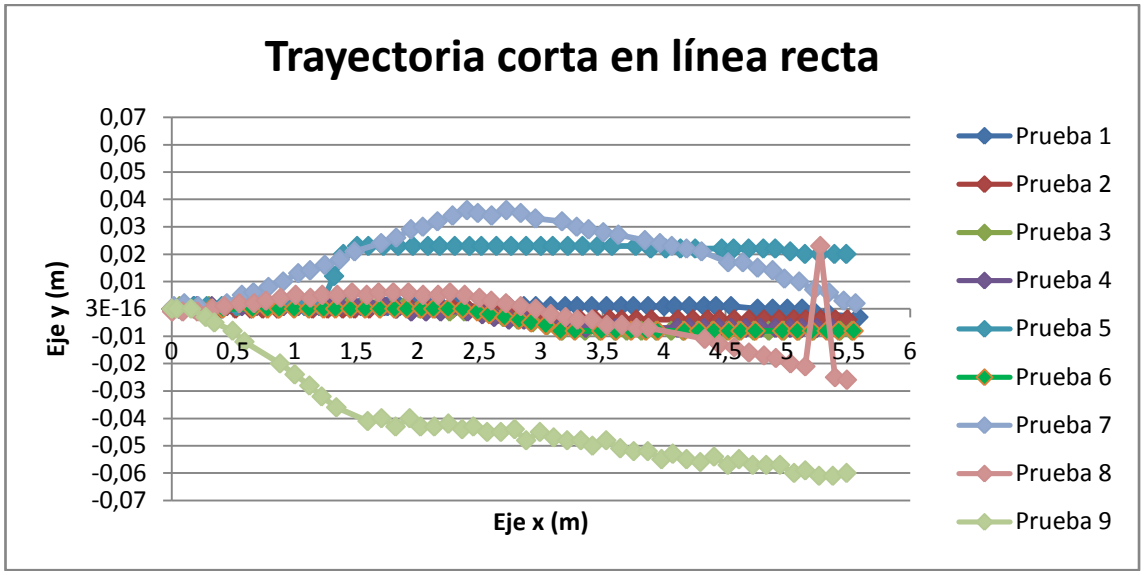



Figura 64. Trayectoria realizada por el robot Pioneer 3DX desde el punto de referencia 1 hacia el 2.

Como se puede observar el Figura 64 el Robot Pioneer 3DX no es capaz de realizar una trayectoria netamente recta. El mayor error que presentó el odómetro fue al desviar al robot 6 cm con respecto a la trayectoria que debe seguir el robot, después de haber recorrido una distancia aproximada de 5.5 m. Este error no es considerable para la aplicación, ya que lo que se pretende es que el robot llegue cerca al lugar donde se encuentra la marca destino una vez sea reconocida por medio del sistema de visión; más no a un punto fijo.

En la Figura 65 se muestra la elipse de error de las nueve pruebas realizadas en línea recta para la localización del robot basada en la odometría acoplada al motor. Esta localización hace referencia a la posición en la que se encuentra el robot cuando se detiene, con respecto a una marca de referencia inicial. El robot se detiene una vez ha sido detectado el punto de referencia destino por medio de la cámara web.

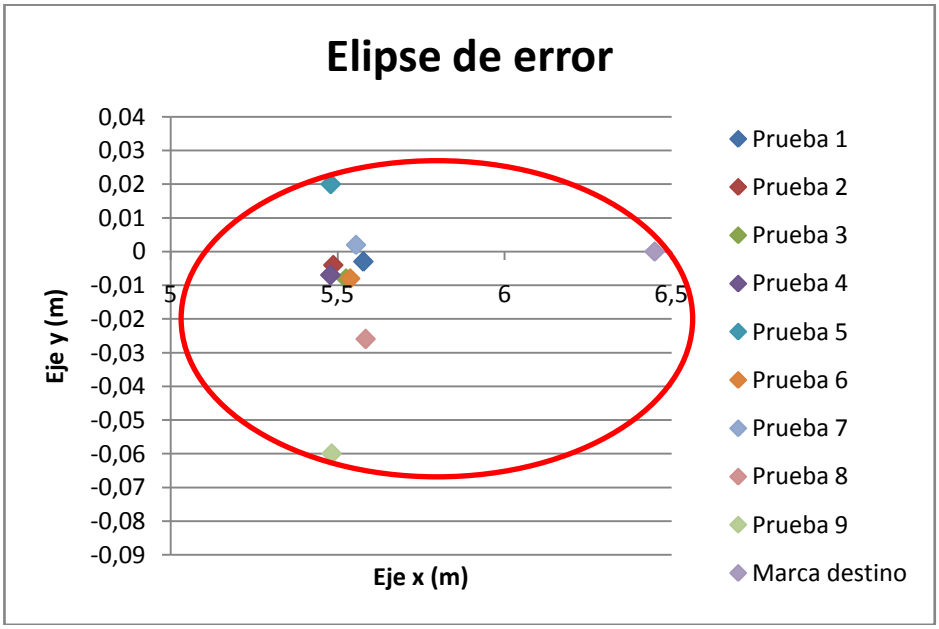



Figura 65. Elipse de error de odometría para una trayectoria corta en línea recta

Como se puede apreciar en la Figura 65 en las nueve pruebas realizadas las posiciones a las que llega la plataforma móvil cuando encuentra la marca destino se encuentran a una distancia casi que despreciable entre ellas. Esto quiere decir que para distancias cortas el odómetro del robot trata de mantener la misma trayectoria desde un mismo punto de partida a un mismo punto final, con lo que se puede afirmar que la incertidumbre para esa distancia es de 10.4 cm en el eje x y de 6 cm en el eje y, lo que incrementa la probabilidad de llegar a la marca destino.

Por otro lado cabe anotar que en las nueve pruebas realizadas se detectó la marca destino. La detección se realizó entre 90 cm y 1 m de distancia entre la cámara y la marca. A esta distancia se detiene el robot al reconocer la marca destino.

En la Figura 66 se muestra cómo se comporta la velocidad del robot para la prueba 1.

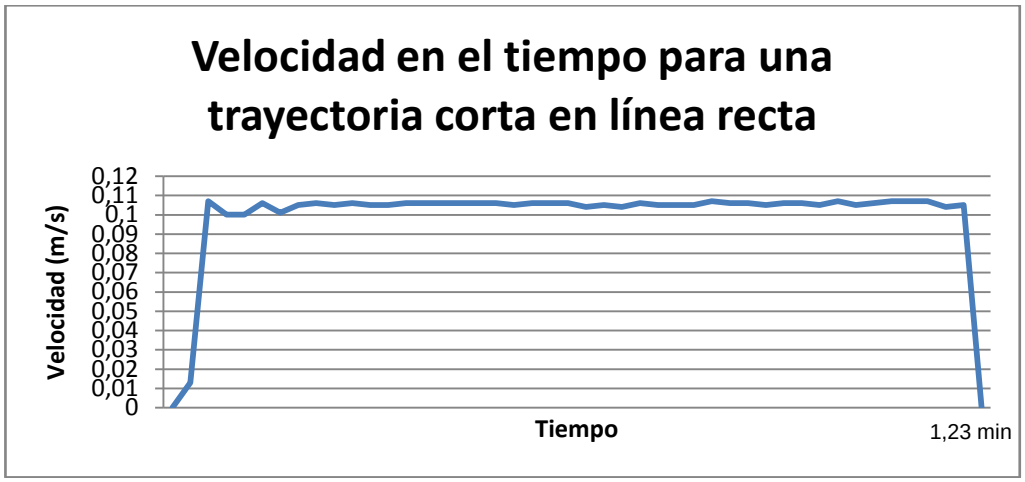



Figura 66. Comportamiento de la velocidad del Robot Pioneer 3DX para la prueba 1.

En la Figura 66 se encuentra que la velocidad alcanzada por el robot se puede considerar constante durante todo su recorrido, ya que las variaciones están en el orden de milésimas.

En la Figura 67 se muestra la ruta generada por el módulo navegador para desplazarse a la marca 23 desde la marca 26.

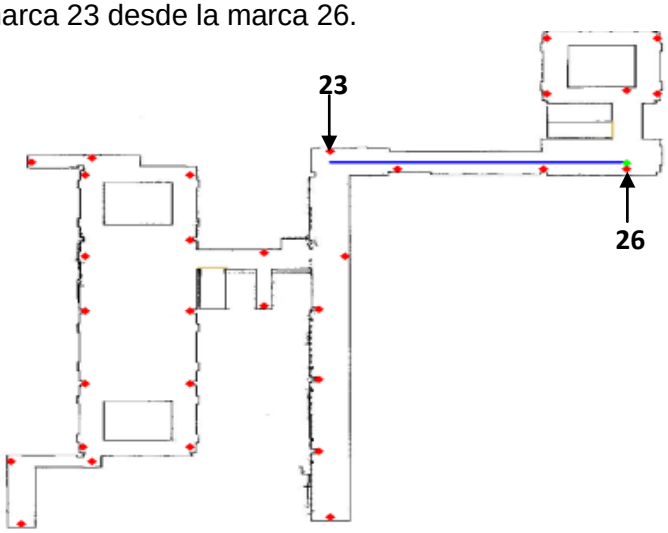



Figura 67. Mapa Topológico con la trayectoria generada por el módulo de navegación para desplazarse de la marca 26 a la 23.

A continuación se muestra la trayectoria descrita por el robot:

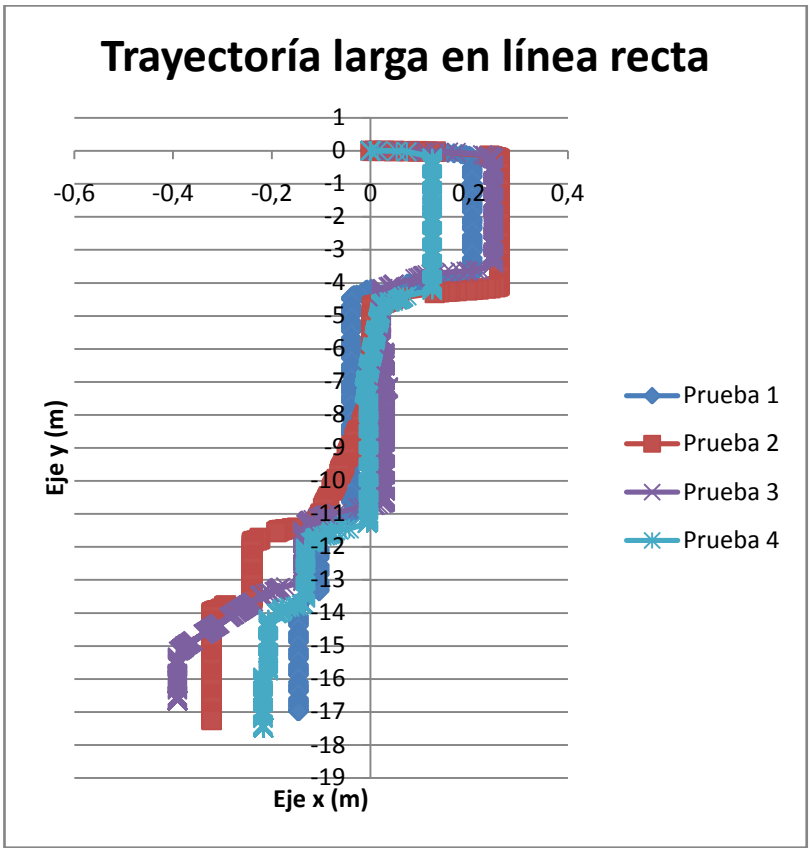



Figura 68. Trayectoria realizada por el robot Pioneer 3DX desde el punto de referencia 26 hacia el 23.

De acuerdo a la Figura 68 se puede observar que el robot trata de seguir la ruta planeada por el módulo navegar descrita en le Figura 67. Las curvas que se visualizan en esta figura es debido a que durante el trayecto el robot encontró obstáculos por medio del anillo de sonares y lo que hace es alejarse de ellos. Entre estos están las columnas del pasillo y algunas paredes. Otra causa de estas curvas realizadas por el robot, es que durante las pruebas había circulación de personas, y estas al pasar cerca al robot, se detectaban como obstáculos y el robot trataba de evadirlos. Inicialmente se produce una curva debido a que el robot esta posicionado de a cuerdo a la Figura 49 y para dirigirse a la marca 23 tiene que hacer un giro de 90°. En la realización de estas pruebas se notó que durante el trayecto recorrido por el robot, éste tiende a desviarse hacia la derecha, lo cual para una distancia recorrida de 17 m el robot se ha desviado aproximadamente 20 cm. Esta desviación no afectó el cumplimiento de la misión, ya que el robot al llegar a la marca destino (marca 23), esta siempre se encontraba en el ángulo de visión de la cámara.

En esta misión, en la primera prueba no se reconoció la marca 25 y en la prueba 2 no se reconoció la marca 24. Esto no afectó la realización de la misión debido a que con saber la marca inicial en que se encontraba el robot y la marca destino, el robot se orientó en relación a estas. En este caso las marcas intermedias solo servían para indicarle al robot que iba por el camino correcto y orientarlo con respecto a la marca destino en caso que se perdiera. Aunque en las dos primeras pruebas no se reconoció toda la secuencia de marcas generadas por el módulo planificador de misión, la misión

se realizó con éxito en las 4 pruebas realizadas, la cual tenía como meta llegar a la marca 23 y fue reconocida en las 4 pruebas.

### 5.2.2 Trayectoria en forma de “L”

En este apartado se procede a implementar una misión donde la trayectoria que debe recorrer la plataforma móvil tenga forma de “L” desde una marca inicial a otra final. Esta prueba se realizó 6 veces para una trayectoria corta y dos veces para una trayectoria larga. El propósito de estas es observar si el odómetro es capaz de representar el movimiento recto y el giro brusco exactamente, o si por el contrario comete un error en la medición y en ése caso poder cuantificarlo.

La primera misión a realizar es que el robot sea capaz de llegar al punto de referencia 15 iniciando desde el 11 siguiendo la ruta generada por el módulo navegador como se muestra en la Figura 69.

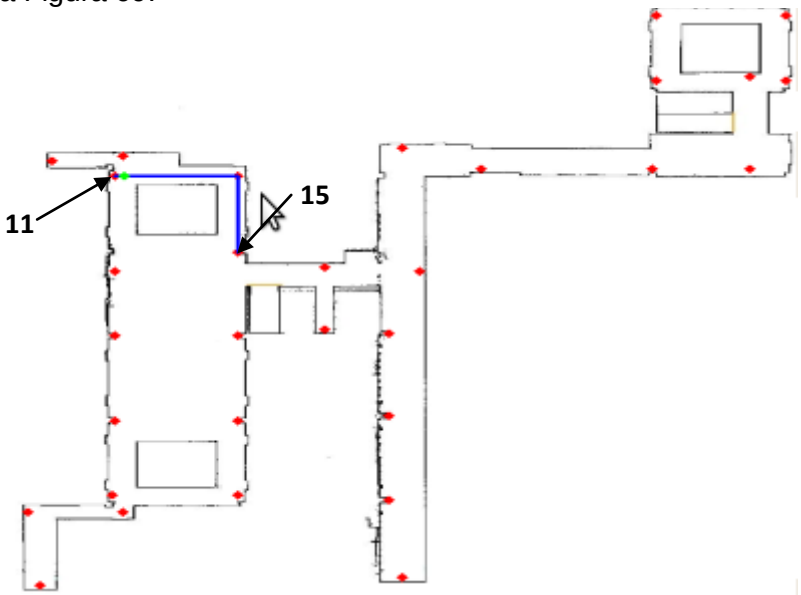



Figura 69. Mapa Topológico con la trayectoria generada por el módulo de navegación para desplazarse de la marca 11 a la 15.

A continuación se muestran las siete pruebas de la trayectoria descrita por el robot para realizar esta misión:

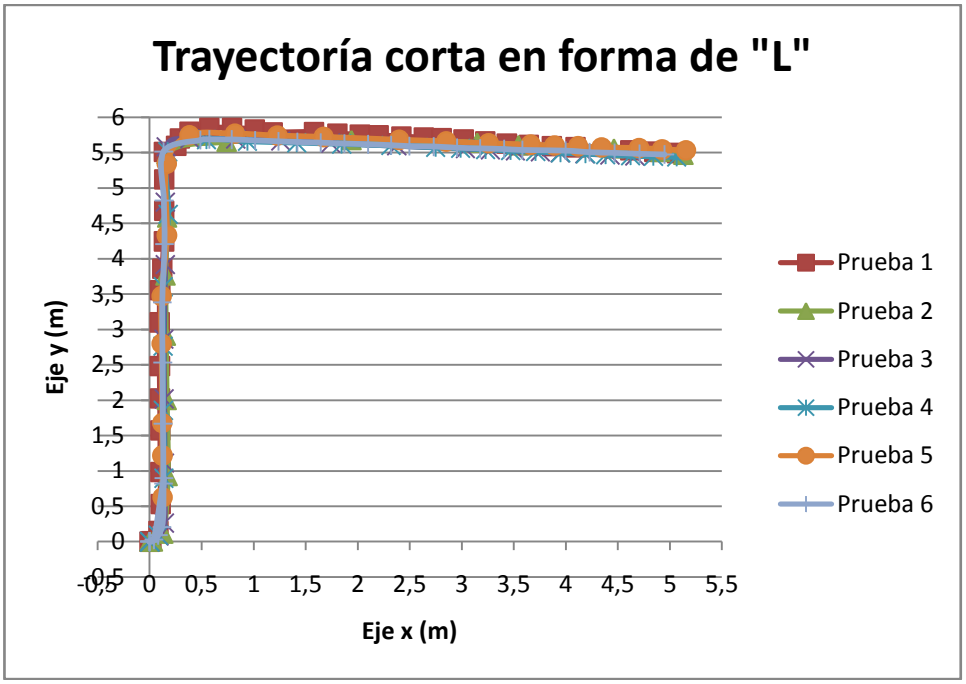



Figura 70. Trayectoria realizada por el robot Pioneer 3DX desde el punto de referencia 11 hacia el 15.

Como se puede observar en las Figuras 70 en las 6 pruebas realizadas la plataforma móvil Pioneer 3DX realiza la misión con éxito describiendo un trayectoria en forma de "L".

Con esta prueba se puede ver el funcionamiento del método de navegación topológica implementado. Este método es el método relacional donde el entorno, que en este caso es la escuela de Ingeniería Eléctrica y Electrónica, se representa como un grafo o como una red de nodos y aristas. Los nodos representan cada una de las marcas o metas que el robot debe reconocer por medio de la cámara web durante su navegación y las aristas son la ruta navegable entre dos marcas generada por el módulo de navegación, las cuales tienen una relación a nivel del espacio. Inicialmente se coloca el robot frente a la marca 11 de acuerdo a la Figura 49. Al iniciar el sistema se realiza la conexión con el Pioneer 3DX y se indica en la GUI la marca inicial en la que se encuentra el robot (en este caso 11) y la marca destino (en este caso 15). Una vez ingresadas ambas marcas el módulo planeador misión genera la secuencia de marcas que se deben reconocer durante la navegación. Luego el módulo navegador genera la ruta entre marca y marca que debe seguir el robot hasta alcanzar la meta. Esta navegación se lleva a cabo por medio del algoritmo VFH dándole al robot una consigna de 10 m en la dirección en la que se encuentra la marca 14. El robot procede a ejecutar dicha trayectoria por medio del módulo piloto dándole el comportamiento apropiado proporcionado por el módulo comportamientos (Moverse hacia el punto (0;10) con respecto a la posición en que se encuentra). Una vez el sistema de visión reconoce la marca 14 el módulo navegador le dará al robot la siguiente consigna de 10 m en dirección a la marca 15. El Robot ejecuta esta trayectoria por medio del módulo piloto seleccionando los comportamientos encontrar referencia, y parar una vez encontrada la marca destino.

En la Figura 71 se muestra la elipse de error de las seis pruebas realizadas para una trayectoria corta en forma de "L".

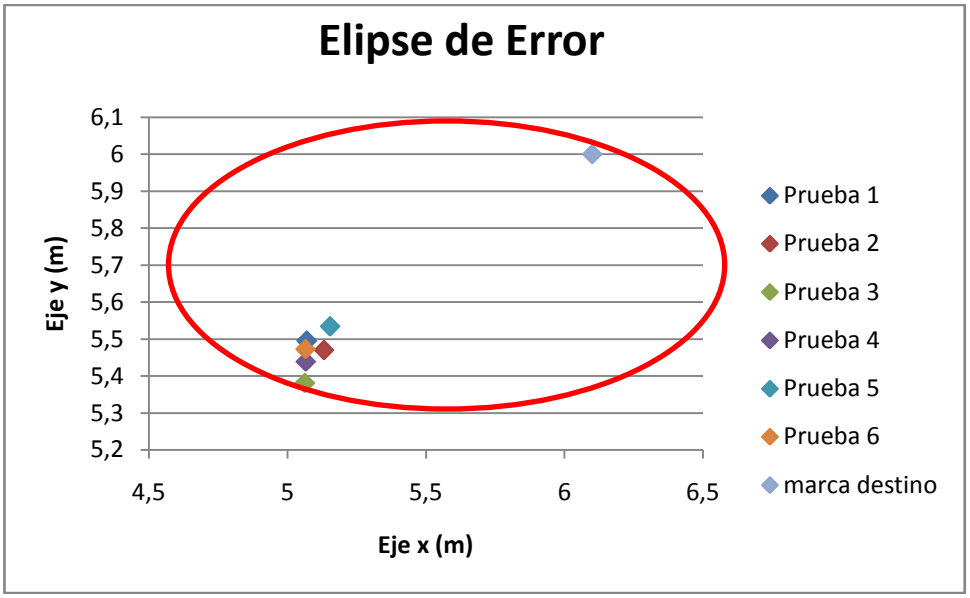



Figura 71. Elipse de error de odometría para una trayectoria corta en línea recta

Como se puede apreciar en la Figura 71 en las seis pruebas realizadas las posiciones a las que llega la plataforma móvil cuando encuentra la marca destino se encuentran a una distancia casi que despreciable entre ellas. De acuerdo a esto se puede afirmar que la incertidumbre en la posición del robot basada en odometría es muy pequeña para una trayectoria corta en forma de "L". Por otro lado se puede ver la eficiencia del sistema de visión, al reconocer la marca destino en las seis pruebas realizadas, a una distancia entre 90 cm y 1 m.

En la Figura 72 se muestra cómo se comporta la velocidad del robot para una trayectoria en forma de "L".


Figura 72. Comportamiento de la velocidad del Robot Pioneer 3DX para una trayectoria corta en forma de "L" para la prueba 4.

Al comparar la velocidad en el tiempo alcanzada por el robot en la Figura 72 con la velocidad mostrada en la Figura 66 se puede observar que la primera presenta más oscilaciones. A pesar de esto se puede considerar que la velocidad aún sigue comportándose constante durante todo el recorrido, ya que las variaciones están en el orden de milésimas.

La segunda misión a realizar es que el robot sea capaz de llegar al punto de referencia 18 iniciando desde el 26 siguiendo la ruta generada por el módulo navegador como se muestra en la Figura 73.

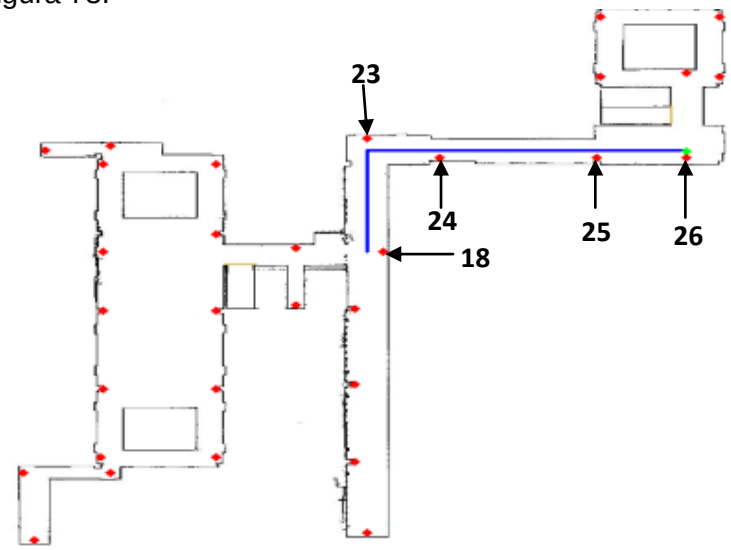



Figura 73. Mapa Topológico con la trayectoria generada por el módulo de navegación para desplazarse de la marca 26 a la 18.

A continuación se muestra la trayectoria descrita por el robot:

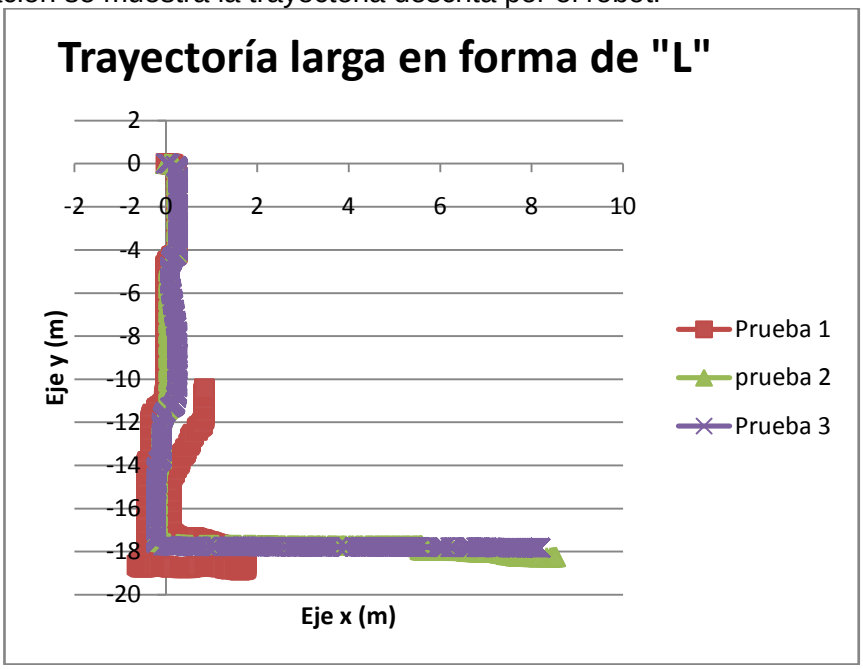



Figura 74. Trayectoria realizada por el robot Pioneer 3DX desde el punto de referencia 26 hacia el 18.

De acuerdo a la Figura 74 se puede observar que en las pruebas 2 y 3 la plataforma móvil Pioneer 3DX realiza la misión con éxito describiendo un trayectoria en forma de "L". En ambas pruebas no se reconoció la marca 24, lo cual no afectó el éxito de la misión, ya que el robot siguió su trayectoria hacia la marca 23, la cual es indispensable para dirigirse hacia la marca 18 debido a que la marca 23 al ser reconocida le proporciona al robot la dirección de movimiento que debe realizar para dirigirse hacia la marca 18 (en este caso un giro de 90°). En caso que la marca 23 no hubiera sido reconocida, el robot hubiera seguido navegando de forma aleatoria como sucede en la prueba 1, hasta que hubiera reconocido otra marca del entorno, y a partir de esta poder orientarse y volver a planear la ruta hacia la marca destino. Con respecto a la prueba 1, a parte de lo explicado la misión no se logró realizar con éxito, debido a que se produjo un error en la ejecución del programa cliente por falta de memoria **("Error: OutOfmemory").** Este error se logró solucionar empleando algunas funciones de las librerías de OpenCV para liberar la cabecera y los datos de las imágenes, por medio de las funciones ***cvFree*** y ***cvReleaseImage().***

La marca 24 no fue reconocida en las pruebas debido a que el lugar donde se realizó la prueba, presenta cambios de iluminación, lo cual afecta el subsistema de visión.

### 5.2.3 Trayectoria en forma de "U"

En este apartado se procede a implementar una misión donde la trayectoria que debe recorrer la plataforma móvil tenga forma de "U" desde una marca inicial a otra final, donde el robot debe realizar dos giros de 90 grados. Esta prueba se realizó con el propósito de observar si el odómetro es capaz de representar los movimientos rectos y los giros exactamente, o si por el contrario comete un error en la medición y en ése caso poder cuantificarlo.

La misión a realizar es que el robot sea capaz de llegar al punto de referencia 16 iniciando desde el 1 siguiendo la ruta generada por el módulo navegador como se muestra en la Figura 75.

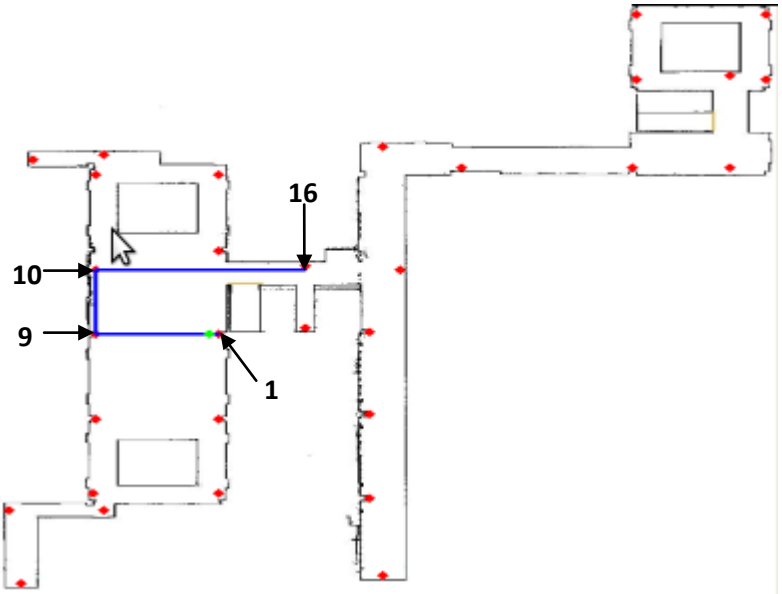



Figura 75. Mapa Topológico con la trayectoria generada por el módulo de navegación para desplazarse de la marca 1 a la 16.

A continuación se muestran la trayectoria descrita por el robot para realizar esta misión:

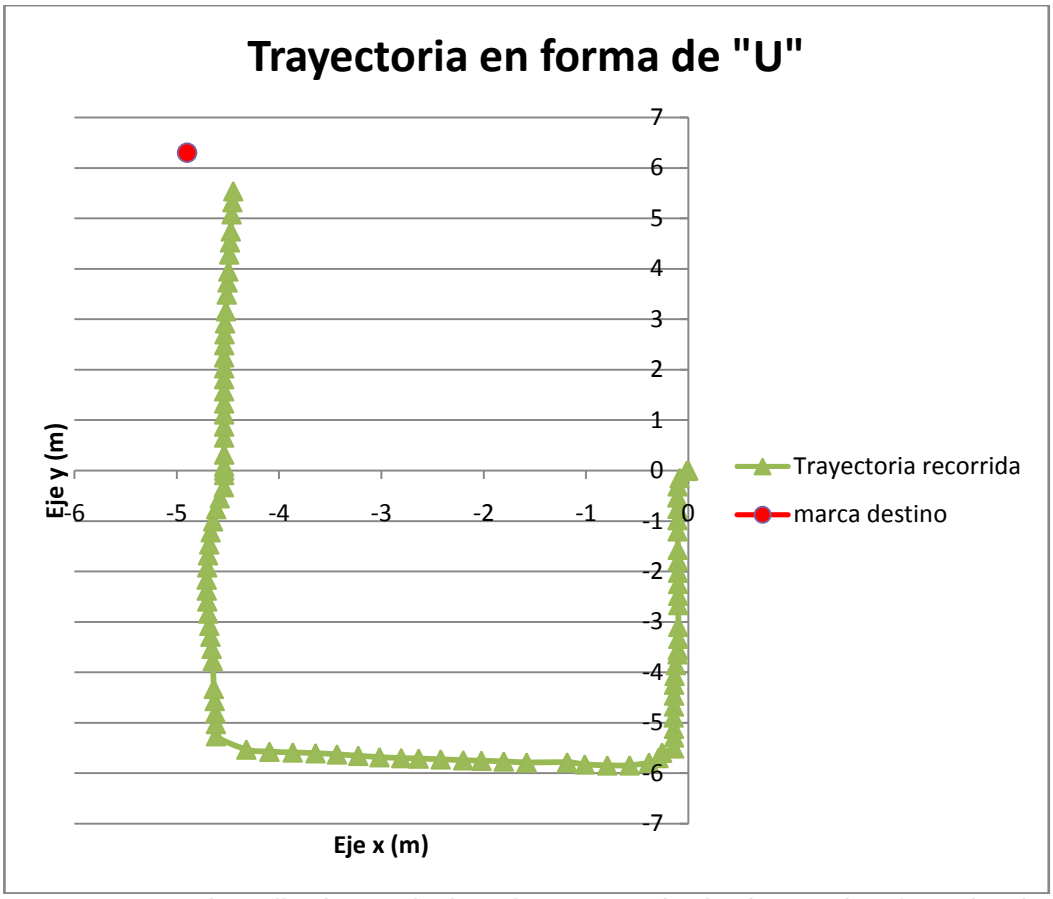



Figura 76. Trayectoria realizada por el robot Pioneer 3DX desde el punto de referencia 1 hacia el 16.

En la Figura 76 se puede observar que la plataforma móvil describe una trayectoria en forma de “U” para llegar a su al punto de referencia destino (marca 16), así como la generó el módulo de navegación. A pesar que el robot tuvo errores de odometría al desviarse 16 cm hacia la derecha después de haber realizado dos giros de 90° y haber recorrido una distancia aproximada de 11.3 después del último giro, la misión se logró con éxito, ya que la marca destino fue reconocida por el sistema de visión. Este marca fue reconocida a una distancia aproximada de 97 cm. Por otro lado se puede observar la eficiencia del sistema de visión para lugares con iluminación controlada al haber reconocido la secuencia de marcas generada por el módulo de planeador de misión, para realizar la misión con éxito.

### 5.3 CONCLUSIONES

El procedimiento de validación permitió corroborar el correcto funcionamiento de cada uno de los módulos principales que componen el sistema de navegación topológica implementado: los módulos de procesamiento de imágenes y reconocimiento de marcas y los módulos de control reactivo/deliberativo.

El conocimiento a priori del mapa topológico fue una gran ventaja ya que al situar el robot en una marca (nodo), éste sabía cómo dirigirse hacia la otra hasta poder llegar a su meta.

Los módulos de procesamiento de imágenes y reconocimiento de patrones permiten una identificación satisfactoria del 100% para una distancia de 90 cm, en condiciones de iluminación controlada (interiores); y del 72,1% para una distancia de 90 cm en exteriores.

El subsistema de visión presenta ciertas limitaciones entre estas se encuentra el ángulo de visión de la cámara, ya que éste al ser muy pequeño (de 60°) no abarca el entorno suficiente para detectar marcas que se encuentran por fuera de éste ángulo, lo que conlleva a que el robot pueda perderse.

Se definió una distancia máxima de 6 m entre marcas ya que a partir de esta distancia el error odométrico es de 6 cm, lo que conllevaría a que el robot pueda perderse y al no seguir la trayectoria planeada se reduce la posibilidad que las marcas se encuentren dentro del ángulo de visión de la cámara. De acuerdo a las pruebas realizadas se concluye que para distancias en línea recta menores a 6 m recorridas por el robot móvil, el error odométrico no afecta la aplicación, lo cual permite que el robot móvil puede desplazarse de un punto de partida conocido y alcanzar un punto de llegada denominado objetivo de manera confiable para trayectos menores a 6 m entre marca y marca.

En base a las pruebas se concluye que al no tener una mapa global del entorno para aplicar técnicas de corrección de odometría, hace que el sistema de navegación topológica implementado no sea lo suficientemente eficiente en trayectorias largas entre marca y marca, ya que entre más distancia recorra la plataforma el error odométrico incrementa, lo que produce que la desviación del robot sea muy grande y no se completen con éxito las misiones a realizar.

Al no tener un sistema de localización global y solo utilizar información local como odometría, si la identificación de las marcas del entorno no se llevan a cabo el robot puede perderse. Para esto el sistema tiene la ventaja que la plataforma así se pierda, tiene la posibilidad que en su recorrido encuentre otra marca y partir de esta el módulo planeador de misión generará una nueva secuencias de marcas que deberán ser reconocidas. Con base a estas, el módulo navegador generará una nueva trayectoria para poder llegar a la marca destino indicada inicialmente por el usuario desde la GUI.

Con las pruebas realizadas se midió el alcance del sistema de navegación topológica implementado para diferentes tipos de trayectorias (en línea recta, en forma de “L”, en forma de “U”) las cuales se llevaron a cabo de forma satisfactoria, tienen muy buenos resultados al cumplir con todas las misiones propuestas para dirigir al robot desde una marca inicial a otra final. De acuerdo a los resultados presentados, en la mayoría de las pruebas el robot llegó con éxito a su destino aunque se presentaron ciertos inconvenientes en el reconocimiento de marcas para lugares con iluminación variable, lo que hace que el sistema presente ciertas limitaciones.

# CAPITULO VI

## 6 CONCLUSIONES GENERALES

La implementación de un sistema de navegación topológica abre un nuevo campo de investigación en el área de la robótica móvil dentro del grupo de investigación Percepción y Sistemas Inteligentes (PSI) de la Universidad del Valle, ya que se puede tomar como punto de partida para realizar investigaciones en áreas tales como: navegación deliberativa, aprendizaje, identificación de patrones, procesamiento de imágenes, etc.

Se realizó un estado del arte de los diferentes métodos utilizados para implementar la navegación topológica, del cual se escogió como método a implementar la navegación topológica relacional por ser una de las técnicas más utilizadas, donde se representa el entorno como un grafo o como una red de nodos y aristas. Los nodos representan gateways, landmarks o metas. Las aristas representan una ruta navegable entre dos nodos, los cuales tienen una relación a nivel del espacio. Esta técnica favoreció la implementación del sistema, ya que esta permitía tener un conocimiento a priori de la ubicación de una marca con respecto a la otra, lo que facilitó llevar a cabo el proceso de navegación del robot.

Se documentaron las técnicas de procesamiento de imágenes más utilizadas en la navegación topológica de las cuales se implementó el algoritmo *match template* por medio de las librerías de OpenCV utilizando el método *CV_TM_CCOEFF_NORMED,* obteniéndose con éste los mejores resultados para el reconocimiento de marcas.

Se implementó un sistema de navegación topológica relacional usando información visual para llevar a cabo la navegación. Este sistema está basado en una arquitectura de control híbrida reactiva/deliberativa lo que facilita la tarea de navegación ya que cada subsistema se encarga de una tarea en particular: El subsistema de visión es el encargado de adquirir la información proporcionada por el sensor de visión, procesarla y compararla con una base de datos de puntos de referencia del entorno y enviar los resultados al subsistema deliberativo. En este último se encuentran aquellas tareas que requieren tiempo de computo elevado como consecuencia de razonamientos de alto nivel, como lo son: la planificación de secuencia de acciones que llevan al robot de un evento inicial a otro final, de acuerdo a un criterio de optimización; la navegación que convierte esas secuencias en movimientos reales supervisando todo el proceso; y por último, la relocalización que permite situar de nuevo al robot en el entorno. En el subsistema reactivo se encuentran las tareas que interactúan con los sensores y actuadores del robot encargándose el movimiento de éste y de activar los sensores necesarios para la percepción y evasión de obstáculos en el entorno local y alcanzar metas puestas por el nivel deliberativo. Para llevar a cabo el funcionamiento del subsistema reactivo se utiliza el algoritmo VFH, el cual utiliza un descriptor basado en el algoritmo de campos de potencial apoyado en técnicas reactivas.

Durante la elaboración del proyecto se utilizó el software Player para comunicarse y acceder al hardware del robot Pioneer 3DX, además sus librerías facilitaron la elaboración de aplicaciones propias para el control del robot. Para el procesamiento de imágenes se utilizaron las librerías de OpenCV lo que permitió implementar los algoritmos necesarios para el procesamiento de imágenes capturadas del entorno de navegación y para el reconocimiento de marcas.

Se desarrolló una interfaz gráfica de usuario (GUI) que le permite interactuar fácilmente con el sistema de navegación topológica implementado. Por medio de esta el usuario puede conectarse vía Wi-Fi a la plataforma móvil Pioneer 3DX en la cual corre el programa servidor; visualizar el entorno por donde navega el Robot, así como el mapa topológico construido del segundo piso de la Escuela de Ingeniería Eléctrica y Electrónica con la ubicación cada uno de los puntos de referencia. La interfaz también posee dos paneles que permiten indicarle al robot la marca inicial en que se encuentra y la marca destino a la cual se desea que navegue el robot. Una vez el usuario ingrese ambas marcas, se mostrará en el mismo mapa topológico la ruta a seguir desde la marca donde se encuentre el Robot hasta la marca destino; como también la posición en la que se encuentre el Robot, una vez reconozca la marca. Aparte de esta GUI se utiliza una de las interfaces proporcionados por el software Player, en la cual se muestra gráficamente la lectura del anillo de sonares del robot; y la odometría y la velocidad del robot en las coordenadas $x,y,\theta$.

El subsistema de visión es fundamental ya que su rapidez y precisión influyen notablemente en el desempeño de la navegación del robot para desplazarse de una marca inicial a otra final. Por esta razón se realizaron las correcciones debidas a la distorsión provocada por la lente y se implementó técnicas de procesamiento para mejorar la imagen y se trabajó en el espacio de color HSV para hacerlo independiente a pequeños cambios de iluminación.

El conocimiento a priori del mapa topológico es una ventaja, ya que por medio de éste es más fácil planificar trayectorias y llevar a cabo la navegación basándose en características del entorno y sus relaciones. Por lo tanto, este mapa va a estar constituido por todas las indicaciones posibles para llegar desde cualquier sitio a otro lugar dentro de un entorno. Más que una sucesión de elementos del entorno, este mapa está formado por una sucesión de tareas a realizar durante la navegación, es decir todos los planes posibles.

# CAPITULO VII

## 7 TRABAJOS FUTUROS

En el Sistema robótico de navegación topológica implementado pueden llevarse a cabo modificaciones tendientes a mejorar o a incursionar en otros campos investigativos.

Se propone utilizar otro tipo de estrategias de control desarrollando otras habilidades deliberativas como:

- Modelado del entorno. Con esta habilidad se pretende que el robot vaya generando un mapa del entorno y actualizando el mapa topológico en caso de que sea necesario durante la navegación, que permita posteriormente al robot moverse libremente dentro de dicho entorno desde cualquier punto de referencia.

- Generación automática del mapa topológico. Con esta habilidad el robot debe ser capaz de realizar las acciones de movimiento y sensado necesarias para explorar el entorno de navegación y generar el mapa topológico sin información a priori.

- Relocalización topológica. Mediante esta habilidad el robot debe de resituarse dentro del mapa topológico en el caso de que se detecte que el robot se ha perdido.

Se propone utilizar otro tipo de técnicas de navegación local utilizando la cámara como sensor para evadir obstáculos. El robot realizaría su recorrido predeterminado y en el momento que en el ángulo de visión de la cámara entrara un objeto que no estaba definido en el mapa, se recalcule el camino para llegar a la marca destino evadiendo este.

Se propone utilizar un mapa global del entorno que contenga información de la ubicación de cada una de las marcas del entorno y así poder implementar técnicas de localización, donde cada vez que el robot reconozca una marca por medio de su sistema de visión, permita ejecutar correcciones de odometría de éste y se obtenga una localización mucho más precisa de éste con respecto al mapa del entorno.

Una posible ampliación del sistema sería la utilización de la información obtenida por la visión por computador para la navegación cooperativa entre robots. La idea sería utilizar un robot, que tiene incorporada una cámara, para indicar a otros robots rutas y correcciones o bien, equipando varios robots con cámaras digitales, conseguir que naveguen en formación a partir de la información de las marcas de referencia.

Se propone usar dos cámaras para llevar a cabo la navegación con lo cual se puede obtener información de distancia de los objetos del entorno y utilizar estos datos para el reconocimiento de marcas y para la navegación del robot.

Se propone para el sistema implementado emplear cámaras omnidireccionales, estudiar posibles adaptaciones de las técnicas y comparar los resultados obtenidos. Con esto se espera una mejora en la navegación, ya que una cámara omnidireccional captura imágenes que abarcan 360°. La información de las imágenes omnidireccionales resulta más completa por abarcar todo el entorno del robot.

Se propone complementar el mapa topológico con información natural del entorno. Para el sistema implementado se podría incluir el reconocimiento de pasillos, puertas, columnas, etc.

## 8. BIBLIOGRAFÍA


**[ARKIN 89]** Arkin R, onald C. “Motor Schema based Mobile Robot Navigation”, mt.J. Robotics Res., vol. 8, no. 4,pp.99-l 12. Aug. 1989.

**[ARKIN 92]** Arkin R, .C. “AURA: A Hybrid Reactive/Hierarchical Robot Architecture”, Workshop on Architectures for Inteligent Control Systems, IEEE mt.Conf, on Robotics and Automation, Nice, France. May 1992.

**[BAILEY 01]** T. Bailey and E. Nebot. Localization in large-scale environment. Robotics and Autonomous System, 37:261-281, 2001.

**[BENGTSSON 03]** Bengtsson, O. and Baerveldt, A. (2003). Robot localization based on scanmatching - estimating the covariance matrix for the IDC algorithm. International
Journal Robotics and Autonomous Systems, 1{2(44):131-150.

**[BORDS 00]** A.G. Bors and I. Pitas. Prediction and tracking of moving objects in image sequences. *IEEE Trans. Image Processing*, 9(8):1441–1445, 2000.

**[BORENSTEIN 89]** Borenstein J. and Koren Y. (1989). Real-Time obstacle avoidance for fast mobile Robots. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5, pp. 1179-1187, September-October.

**[BORENSTEIN 91]** J. Borenstein and Y. Koren. “The Vector Field Histogram – Fast obstacle avoidance for mobile robots”, IEEE Journal of Robotics and Automation Vol. 7, No 3, June 1991, pp. 278-288.

**[BORENSTEIN 96]** Borenstein, J., Everett, B., and Feng, L. (1996). Navigating mobile robots: systems and techniques. A. K. Peters.

**[BROOKS 86]** Brooks, R.A., (1986). A robust layered control architecture for a mobile
robot. IEEE Journal of Robotics and Automation, 2, (1), 14-23.

**[BURGARD 98]** Burgard, W., Derr, A., Fox, D., and Cremers, A. B. (1998). Integrating global position estimation and position tracking for mobile robots: the dnamic markov localization approach. In Proceedings of the IEEE/RSJ International Conference on Intelligent RObot and Systems.

**[CASTLEMAN 96]** K.R. Castleman. *Digital Image Processing*. Prentice-Hall, Englewood Cliffs, New Jersey, 1996.

**[CHELLAPPA 84]** R. Chellappa and R. Bagdazian. Fourier coding of image boundaries. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-6(1):102–105, 1984.

**[DEAN 93]** Dean, T., Bonasso, R.P., “1992 AAAI Robot Exhibition and Competition,” *AI Magazine*, vol. 14, no. 1, Spring 1993, pp. 35–48.

**[DESUOZA 02]** Guilherme N. DeSuoza and Avinash C. Kak, *Vision for mobile robot navigation: A survey*, IEEE Transactions on pattern analysis and machine intelligence **24** (February 2002).

**[DUCKETT 99]** Duckett, T. and Nehmzow, U. (1999). Knowing your place in real world environments. In Proceedings of the Eurobot.

**[ELFES 89]** Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. Computer, 22:46-57.

**[ESPINOSA 08]** Espinosa, F., Salazar, M., Valdes, F., Bocos, A. *“Communication architecture based on Player/Stage and sockets for cooperative guidance of robotic units”* - 16th Mediterranean Conference on Control and Automation. 2008

**[EVERETT 95]** Everett, H. R. (1995). Sensors for mobile robots. Theory and applications. A. K. Peters, Ltd.

**[FAUGERAS 80]** O. Faugeras and W. Pratt. Decorrelation methods of texture feature extraction. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-2(4):323–332, 1980.

**[FRANZ 98]** Franz, M. O., Scholkopf, B., Mallot, H. A., and Bulthoff, H. H. (1998). Where did I take the snapshot? scene-based homing by image matching. Biological
Cybernetics, 79:191-202.

**[FRANZ 00]** M. Franz and H. Mallot. Biomimetic robot navigation. Robotics and Autonomous System 30, 2000.

**[GASPAR 00]** J. Gaspar, N. Winters, and J. Santos-Victor. Vision based navigation and environmental representations with an omni-directional camera. IEEE TRA, 16(6), December 2000.

**[GREINER 94]** Greiner, R. and Isukapalli, R. (1994). Learning to select useful landmarks. In Proceedings of the Twelfth National Conference on Artificial Intelligence, pages 1251-1256. AAAI Press/MIT Press.

**[GVH 03]** Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage project: tools for multi-robot and distributed sensor systems. In Proceedings of the 11th International Conference on Advanced Robotics ICAR'2003, pages 317–323, Coimbra (Portugal), June 2003.

**[HARALICK 79]** R.M. Haralick. Statistical and structural approaches to texture. *Proc. IEEE*, 67(5):786–804, 1979.

**[HARTMANN 96]** I. Hartmann. *Mustererkennung*. Skriptreihe Regelungstechnik und Bildverarbeitung, Technische Universit¨at Berlin, 1996.

**[IWAN 00]** Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. Proceedings IEEE International Conference on Robotics and Automation, San Francisco, pages 1023-1029, April 2000.

**[JÄHNE 95]** B. Jähne. *Digitale Bildverarbeitung*. Springer, Berlin, Heidelberg, 2 edition, 1995.

**[JAIN 00]** A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.
**[JAN 99]** Jan Puzicha, Joachim M. Buhmann, Yossi Rubner, and Carlo Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In Proceedingsthe IEEE International Conference on Computer Vision (ICCV1999), 2:1165 – 1173, 1999.

**[KAMM 98]** K.-F Kamm. Grundlagen der R.ontgenabbildung. In K. Ewen, editor, *Moderne Bildgebung: Physik, Ger¨atetechnik, Bildbearbeitung und -kommunikation, Strahlenschutz, Qualit¨atskontrolle*, pages 45–62, Stuttgart, New York, 1998. Georg Thieme Verlag.

**[KOREN 91]** Koren Y. and Borenstein J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California, April.

**[KORTENKAMP 94]** Kortenkamp,D., and Weymouth, T., "TopologicalMapping for Mobile Robots using a Combination of Sonar and Vision Sensing," proceedings *AAAI-94*, 1994, pp. 974–984.

**[KUIPERS 87]** Kuipers, B.J. y Byun, Y.T. "A qualitative approach to robot exploration and map-learning. Spatial Reasoning and Multi-Sensor Fusion: Prooceedings of the 1987 Workshop. p 390-404. Morgan Kaufmann. USA. 1987

**[KUIPERS 91]** Kuipers, B, and Byun, Y.T., "A Robot Exploration and Mapping Strategy Based on a Semantic Hierarchy of Spatial Representations," *Robotics and Autonomous Systems*, vol. 8, 1991, pp. 47-63.

**[LAMBRINOS 00]** Lambrinos, D., Moller, R., Labhart, T., Pfeifer, R., andWehner, R. (2000). A mobile robot employing insect strategies for navigation. Robotics and Autonomous Systems. Speciall issue on biomimetic robot, 30:39-64.

**[LEVITT 90]** Levitt, T.S., and Lawton, D.T., "Qualitative Navigation for Mobile Robots," *AI Journal*, vol. 44, no. 3, Aug 1990, pp. 305-360.

[**LOCCHI 03**] IOCCHI, Luca; NARDI, Daniele y SALERNO, Massimiliano. Reactivity and Deliberation: a survey on Multi-Robot Systems. Italia, 2003.

**[LOPEZ 05]** López Romero, A.: "Navegación Global utilizando grafo de visibilidad.", Proyecto Fin de Carrera, Universidad Rey Juan Carlos, 2005.

**[MARR 80]** D. Marr and E. Hildreth. Theory of edge detection. *Proc. Roy. Soc. London*, B(207):187–217, 1980.

**[MARTIN 96]** Martin, M. C. and Moravec, H. (1996). Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute. Carnagie Mellon University.

**[MARTÍNEZ 97]** Muñoz Martínez, Victor Fernando. Planificación de trayectorias para robots móviles, 208 páginas. Departamento de Ingeniería de Sistemas y Automática de la Universidad, 1997.

**[MERY 01]** D. Mery. *Automated Flaw Detection in Castings from Digital Radioscopic Image Sequences*. Verlag Dr. K¨oster, Berlin, 2001. (Ph.D. Thesis in German).

**[MORENO 03]** Moreno, L. and Dapena, E. (2003). Path quality measures for sensor-based motion planning. International Journal Robotics and Autonomous Systems, 1- 2(44):131-150.

**[MRT 03]** Michael Montemerlo, Nicholas Roy, and Sebastian Thrun. Perspectives on standardization in mobile robot programming: the Carnegie mellon navigation (CARMEN) toolkit. In Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03), volume 3, pages 2436–2441, Las Vegas (USA), October 2003.

**[MUÑOS 95]** Muñoz M, Victor F., Planificación de Robots para Robots Móviles, Tesis Doctoral, Universidad de Málaga, 1995.

**[MUÑOZ 99]** Victor Fernando Muñoz Martínez, "Navegación en Robots Móviles", Universidad de Málaga, 1999.

**[MURPHY 00]** Murphy, R.R., (2000). Introduction to AI Robotics. MIT Press.

**[NEHMZOW 00]** Nehmzow, U. and Owen, C. (2000). Experiments with manchester's fourty two in unmodi_ed large environments. Robotics and Autonomous Systems, 33:223-242.

**[NEHM 95]** Nehmzow U, . "Animal and robot navigation." Robotics and Autonomous Systems. Vol. 15, Iss. 1-2, p.7l-8l. Netherlands. 1995.

**[NII 89]** Nii, H. P., (1989). Introduction, in: Blackboard architectures and applications. Edited by V. Jagannathan, Rajendra Dodhiawala and Lawrence S. Baum, (Perspectives in artificial intelligence, volume 3). Academic Press, Boston, pp. xix-xxix.

**[NILSON 69]** Nilson, N, J, (1969) A Movile Automaton: An Application of Artificial Intelligence Techniques, Procceding of the first International Joint Conference on Artificial Intelligence, Washington D.C

**[OHM 95]** J.-R. Ohm. *Digitale Bildcodierung*. Springer, Berlin Heidelberg, 1995.

**[OLLERO 01]** OLLERO, Aníbal. Robótica. Manipuladores y robots móviles. España: Marcombo Editores, 2001. 447 p.

**[OPENCV]** http://www.sourceforge.net/projects/opencvlibrary

**[OWEN 98]** Owen, C. and Nehmzow, U. (1998a). Landmark-based navigation for a mobile robot. In Simulation of Adaptive Behavior.

**[PERSOON 77]** E. Persoon and K.S. Fu. Shape discrimination using fourier descriptors. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-7(3):170–179, 1977.

**[PLAYER/STAGE]** Documentación de Player/Stage http://playerstage.sourceforge.net

**[POPESCU 03]** Popescu, N. (2003). Robust self-localization of a robot by intelligent fuzzy system. In 14th Conference on Control Systems and Computer Science, pages 175-179.

[**RANDAL 97**] Randal C. Nelson, "From Visual Homing to Object Recognition", in *Visual Navigation* Yiannis Aloimonos, Editor, Lawrence Earlbaum, 1997, 218-250.

**[RIZZI01]** Rizzi, A., Duina, D., Inelli, S., and Cassinis, R. (2001). A novel visual landmark matching for a biologically inspired homing. Pattern Recognition Letters, 22:1371-1378.

**[RWI 99]** Real World Interface RWI. Mobility Robot Integration software, User's guide. Technical Report version 1.1, IS Robotics, Inc, May 1999.

**[SIMMONS 95]** Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable eenvironments. In Proceedings of the International Joint Conference on Arti_cial Intelligence, pages 1080{1087.

**[SMITH 86]** Smith, R., and Cheeseman, P., "On the Representation of and Estimation of Spatial Uncertainty," *International Journal of Robotics Research*, vol. 5, 1986, pp. 56–68.

**[SMITH 88]** Smith R., Self M., Cheeseman P. "Estimating uncertain spatial relationships in robotics". Uncertainty in Artificial Intelligence 2. J. F. Lemmer and L. N. Kanal, Eds. New York: Elsevier. 1988.

**[SONKA 98]** M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysys, and Machine Vision*. PWS Publishing, Pacific Grove, CA, 2 edition, 1998.

**[THRUN 98]** Thrun, S. (1998). Bayesian landmark learning for mobile robot navigation. Machine Learning, 33(1):41-76.

**[TRAHANIAS 99]** Trahanias, P. E., Velissaris, S., and Oraphanoudakis, S. C. (1999). Visual recognition of workspace landmarks for topological navigation. Autonomous Robots, 7:143-158.

**[TRULLIER 97]** Thruiller, O.; Meyer, J. A. "Biomimetic navigation models andstrategies in anirnats." AI Communications. Vol. 10, Iss. 2, p.'79-82. IOS Press. Netherlands. 1997.

**[ZAHN 71]** C.T. Zahn and R.Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Trans. Computers*, C-21(3):269–281, 1971.