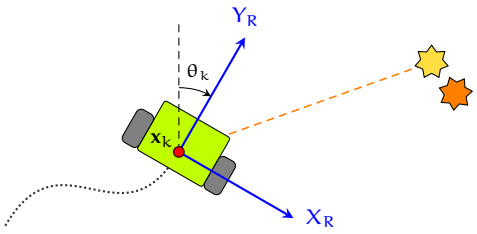
TÉCNICAS DE NAVEGACIÓN PARA UN ROBOT MÓVIL UTILIZANDO SISTEMAS DE RAZONAMIENTO ESPACIAL

CARLOS FERNÁNDEZ CARAMÉS



Tesis Doctoral

Departamento de Informática y Automática
Facultad de Ciencias
Septiembre 2012

VNiVERSiDAD D SALAMANCA
CAMPUS DE EXCELENCIA INTERNACIONAL

Carlos Fernández Caramés: *Técnicas de navegación para un robot móvil utilizando sistemas de razonamiento espacial,* Tesis Doctoral, © Septiembre 2012

DIRECTORES:
Vidal Moreno Rodilla
Belén Curto Diego

LUGAR:
Salamanca

FECHA:
Septiembre 2012

"Un padre vale por cien maestros." — *George Herbert*

A mis padres.

RESUMEN

A la hora de integrar a los robots móviles autónomos en nuestra sociedad, el desafío es conseguir que estén presentes de forma natural en el interior de los edificios, con el fin de que puedan realizar tareas como guiar visitantes, repartir medicinas en hospitales, monitorizar la seguridad de un edificio, asistir a discapacitados o incluso actividades de mayor complejidad. Este tipo de actividades se encuadran no sólo dentro de la robótica autónoma sino también en el campo de la *inteligencia ambiental*, cuyo fin último es el desarrollo de sistemas inteligentes integrados de manera transparente en el entorno y con la capacidad de asistir al ser humano. Además, los robots de servicios suelen ser objeto de interés para grandes corporaciones e instituciones gubernamentales, que hoy en día disponen de sistemas de gestión de inmuebles (*facility management*) donde almacenan todos los datos relacionados con sus instalaciones. Por tanto, sería una necesidad integrar a los robots en estos sistemas, dado que cada vez más los robots se están convirtiendo en un elemento clave como recurso.

Así pues, tanto en un sistema de inteligencia ambiental como en un sistema de gestión de inmuebles, un robot debe estar localizable en todo momento. De este modo puede estar disponible en caso de ser necesario, al igual que cualquier otro recurso de un edificio (medicinas, extintores, empleados). Esto implica que un robot puede ser un recurso activo y disponible en cualquiera de estos dos sistemas siempre y cuando tenga la capacidad de mantenerse localizado en su entorno. Además, la necesidad de localización convierte a un robot no sólo en un recurso sino también usuario de la información espacial almacenada en el sistema, ya que para estimar la posición es imprescindible disponer de un mapa donde ubicarse.

Una de las principales carencias de los sistemas de gestión de inmuebles tradicionales es que tienen una capacidad muy limitada de análisis espacial. Con objeto de afrontar esta necesidad, recientemente se han comenzado a incorporar los Sistemas de Información Geográfica (SIG) en la gestión de inmuebles, ya que poseen una capacidad de análisis espacial muy avanzada. Son, por ejemplo, capaces de manejar al mismo tiempo los datos espaciales de un edificio y relacionarlos con su situación dentro de un campus, una ciudad, una nación o incluso a nivel mundial. Otro aspecto aún más importante es que pueden manejar y almacenar información simbólica (puertas, mesas, o ventanas), lo que los convierte en una herramienta muy útil para poder dotar a los robots de una mayor conciencia espacial, y genera una de las principales necesidades de esta tesis: la extracción de información simbólica y su localización relativa a partir de la información sensorial.

PUBLICACIONES

ARTÍCULOS EN REVISTAS

Fernández-Caramés, C.; Moreno, V.; Curto, B. y Vicente, J. A. "Clustering and line detection in laser range measurements". *Robotics and Autonomous Systems 58*, 5 (2010), 720–726.

Curto, B.; Moreno, V.; Fernández-Caramés, C.; Chehayeb, A. y Alves Santos, R. "Applying a software framework for supervisory control of a plc-based flexible manufacturing systems". *The International Journal of Advanced Manufacturing Technology 48*, 5-8 (2010), 663–669.

CONGRESOS INTERNACIONALES

Fernández-Caramés, C.; Moreno, V.; Curto, B. y Blanco Rodríguez, F. J. "Extracción de características en un agente cartógrafo mediante un escáner láser 2d". En: *Proceedings of the 6th International Workshop on Practical Applications of Agents and Multiagent Systems* (Salamanca, España, 2007), J. Bajo, V. Alonso, L. Joyanes, and J. M. Corchado, Eds., Universidad de Salamanca, pp. 139–148.

Fernández-Caramés, C.; Moreno, V.; Curto, B. y Blanco Rodríguez, F. J. "Clustering and line detection in laser range measurements". En: *Proceedings of the 3rd International Workshop on Intelligent Robotics (IROBOT 2008)* (Oct. 2008), L. P. Reis, L. Correia, N. Lau, and R. Bianchi, Eds., pp. 13–24.

Fernández-Caramés, C.; Moreno, V.; Curto, B. y Vicente, J. A. "Fuzzy trajectory tracking for an autonomous mobile robot". En: *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics, Intelligent Control Systems and Optimization (ICINCO 2009)* (Julio 2009), J. Filipe, J. Andrade-Cetto, and J.-L. Ferrier, Eds., INSTICC Press, pp. 359–362.

Curto, B.; Moreno, V.; Fernández-Caramés, C.; Alves Santos, R. y Chehayeb, A. "Applying a software framework for supervisory control of a plc-based discrete event system". En: *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics, Volume Robotics and Automation* (Julio 2009), J. Filipe, J. Andrade-Cetto, and J.-L. Ferrier, Eds., INSTICC Press, pp. 263–267.

CONGRESOS NACIONALES

Fernández-Caramés, C.; Moreno, V.; Curto, B.; Rodríguez, A. y Vicente, J. A. "Diseño, construcción e integración de un robot móvil autónomo en missionlab". En: *XXX Jornadas de Automática* (Sept. 2009).

Curto, B.; Moreno, V.; Alves Santos, R.; Fernández-Caramés, C. y Chehayeb, A. "Diseño e implementación de un sistema de control supervisor basaco en plcs". En: *XXX Jornadas de Automática* (Sept. 2009).

Rodríguez, A.; Curto, B.; Moreno, V.; González, R. y Fernández-Caramés, C. "Herramienta virtual para la enseñanza de robótica basada en realidad aumentada". En: *XXX Jornadas de Automática* (Sept. 2009).

Fernández-Caramés, C.; Moreno, V.; Curto, B.; Alves Santos, R. y Rodríguez-Aragón, J. "Controlador borroso de seguimiento de segmentos para diversas configuraciones cinemáticas". En: *XXXI Jornadas de Automática* (Sept. 2010).

Rodríguez, A.; Curto, B.; Moreno, V.; Serrano, F. J. y Fernández-Caramés, C. "Tarea de limpieza coordinada mediante un equipo heterogéneo de robots". En: *XXXI Jornadas de Automática* (Sept. 2010).

Rodríguez-Aragón, J.; Curto, B.; Moreno, V.; Fernández-Caramés, C. y Serrano, F.J. "Técnicas de visión artificial para estimar el estado de un robot". En: *XXXI Jornadas de Automática* (Sept. 2010).

artículos en revistas pendientes de aceptación

Fernández-Caramés, C.; Rodríguez-Aragón, J.; Moreno, V.; Curto, B. y Serrano, F. J. "A robust and real-time door detection system for robotic applications in urban buildings". Enviado a: *Robotics and Autonomous Systems (Elsevier)* (2012).

Serrano, F.J.; Curto, B.; Moreno, V.; Rodríguez-Aragón, J. y Fernández-Caramés, C. "Creating a powerful multirobot software control architecture: the complete integration of MissionLab and Carmen". Enviado a: *Autonomous Robots (Springer)* (2012).

Fernández-Caramés, C.; Serrano, F. J.; Moreno, V.; Curto, B.; Rodríguez-Aragón, J. y "A real-time indoor localization approach based on spatial reasoning systems". Enviado a: *Autonomous Robots (Springer)* (2012).

AGRADECIMIENTOS

En primer lugar, me gustaría agradecerle a mis padres su apoyo incondicional en los buenos y malos momentos; sin su ayuda jamás hubiera llegado hasta aquí.

También quiero expresar mi agradecimiento a mis directores, Vidal Moreno y Belén Curto, por su ayuda y consejos en la realización de esta tesis doctoral, y especialmente por ofrecerme la posibilidad de comenzar a trabajar en el campo de la robótica autónoma.

No puedo olvidar darle las gracias a las personas que han realizado importantes aportaciones a esta tesis, en especial a Jesús F. Rodríguez, por su imprescindible ayuda tanto en el desarrollo de los algoritmos de extracción de características como en la obtención de los resultados experimentales; a F. Javier Serrano, por la inestimable ayuda que me ha prestado a la hora de realizar las pruebas experimentales con *Morlaco*; a María José Polo y Carlos Iglesias Blázquez por su ayuda en lo relativo al manejo de las bases de datos espaciales; y también a J. Andrés Vicente y Javier Blanco por echarme siempre una mano con *Morlaco*.

Tampoco puedo olvidar a mis compañeros de despacho: Rodrigo, Juan Carlos y Susana, así como otros compañeros y amigos que he tenido a lo largo de todo este tiempo en el laboratorio de robótica, y con los que he pasado muy buenos momentos: Raúl Álves, Juan Bueno, Teny, Manu, Alber, los gemelos (David y Pablo), Rubén y Aníbal. Gracias también a Rubén, Josué y Laura por los buenos ratos y las risas que hemos compartido en los calurosos meses de verano.

Finalmente, también deseo agradecerle a Noelia su apoyo, compañía y amistad durante muchos años y en algunos de los momentos más difíciles de mi vida, al igual que a otros amigos como Pedro, Pablo, Gonzalo y David, por tener siempre un momento para escuchar mis problemas. Y cómo no, a Vicen por su amistad y sus precios de amigo.

AGRADECIMIENTOS INSTITUCIONALES

Esta tesis doctoral ha financiada por la Consejería de Educación de la Junta de Castilla y León y por el Fondo Social Europeo, al amparo de una Beca de Formación de Personal Investigador (ORDEN EDU/1453/2005 de 28 de octubre).

ÍNDICE GENERAL

1 INTRODUCCIÓN 1

1.1 INTRODUCCIÓN 3
- 1.1.1 Motivación . . . 4
- 1.1.2 Objetivos . . . 6
- 1.1.3 Organización de la tesis . . . 6

1.2 ANTECEDENTES Y ESTADO DEL ARTE 9
- 1.2.1 Localización . . . 9
- 1.2.2 Los SIG en robótica . . . 11
- 1.2.3 Detección de puertas . . . 12
- 1.2.4 Líneas verticales . . . 14

1.3 RESUMEN 17

2 INFORMACIÓN ESPACIAL 19

2.1 ALMACENAMIENTO Y ANÁLISIS DE INFORMACIÓN ESPACIAL 21
- 2.1.1 Sistemas de información geográfica . . . 23
- 2.1.2 Representación de información espacial . . . 23
- 2.1.3 Datos espaciales . . . 25
- 2.1.4 Modelo de Objetos Geométricos . . . 27
- 2.1.5 Bases de datos espaciales . . . 28
- 2.1.6 Procesamiento de datos espaciales . . . 30

2.2 RESUMEN 33

3 ESTIMACIÓN DE SISTEMAS ESTOCÁSTICOS 35

3.1 EL FILTRO DE KALMAN (KF) 37
- 3.1.1 Suposiciones . . . 38
- 3.1.2 Formulación del problema de estimación . . . 40
- 3.1.3 Algoritmo . . . 41

3.2 EL FILTRO EXTENDIDO DE KALMAN (EKF) 43
- 3.2.1 Modelo del sistema . . . 44
- 3.2.2 Modelo de observación . . . 44
- 3.2.3 Algoritmo . . . 44
- 3.2.4 Compensación heurística de los errores . . . 45

3.3 MODELO DEL SISTEMA 47
- 3.3.1 Modelo del sistema . . . 47
- 3.3.2 Vector de estado . . . 47
- 3.3.3 Función de transición de estados . . . 48
- 3.3.4 Ruido del sistema . . . 49

3.4 MODELO DE OBSERVACIÓN 51
- 3.4.1 Sensor virtual . . . 51
- 3.4.2 Especificación del modelo de observación . . . 52
- 3.4.3 Función de observación . . . 52
- 3.4.4 Ruido en las observaciones . . . 53

3.5 RESUMEN 57

4 MODELADO Y CALIBRACIÓN DE SENSORES 59
4.1 MODELADO DE SENSORES 61
4.1.1 Introducción . . . 61
4.1.2 LIDAR . . . 62
4.1.3 Cámara . . . 66
4.1.4 Encoders rotatorios . . . 70
4.2 CALIBRACIÓN 73
4.2.1 Sistemas de referencia . . . 73
4.2.2 Calibración del láser . . . 74
4.2.3 Calibración de la cámara . . . 76
4.2.4 Fusión de datos . . . 81
4.3 RESUMEN 83

5 ENFOQUE PROPUESTO 85
5.1 PERSPECTIVA GENERAL 87
5.1.1 Fusión sensorial . . . 88
5.1.2 Detección de puertas . . . 89
5.1.3 Manejo de datos espaciales mediante un SIG . . . 92
5.1.4 Localización . . . 93
5.2 EXTRACCIÓN DE CARACTERÍSTICAS CON LÁSER 95
5.2.1 Definición y ejemplo de barrido láser . . . 95
5.2.2 Cálculo de la orientación de un pasillo . . . 96
5.2.3 Alineación del barrido láser . . . 99
5.2.4 Detección de paredes . . . 99
5.3 DETECCIÓN DE PUERTAS 103
5.3.1 Cálculo de las regiones de interés . . . 103
5.3.2 Características pseudo-Haar . . . 104
5.3.3 Detección de los laterales de las puertas . . . 106
5.3.4 Detección de puertas . . . 110
5.4 CONSULTAS ESPACIALES 115
5.4.1 Conversión de un plano a datos espaciales . . . 115
5.4.2 Formato de datos espaciales . . . 117
5.4.3 Esquema de la base de datos . . . 117
5.4.4 Consultas PostGIS para la asociación de datos . . . 121
5.5 LOCALIZACIÓN MEDIANTE EKF 123
5.5.1 Predicción de la posición del robot . . . 124
5.5.2 Observaciones . . . 125
5.5.3 Predicción de las observaciones . . . 125
5.5.4 Asociación de datos . . . 126
5.5.5 Corrección de la posición . . . 129
5.6 RESUMEN 131

6 RESULTADOS Y CONCLUSIONES 133
6.1 RESULTADOS DE DETECCIÓN DE PUERTAS 135
6.1.1 Pruebas de detección de puertas . . . 135
6.1.2 Ejemplos de clasificación de puertas . . . 138
6.1.3 Análisis de tiempos . . . 141
6.2 RESULTADOS DE LOCALIZACIÓN 147
6.2.1 Entornos de desarrollo robóticos . . . 147
6.2.2 Diseño de pruebas de localización autónomas . . . 149
6.2.3 Resultados experimentales . . . 150
6.3 CONCLUSIONES Y TRABAJO FUTURO 157
6.3.1 Conclusiones generales . . . 157
6.3.2 Conclusiones específicas . . . 158

6.3.3 Trabajo futuro 160

7 APÉNDICES 163

A ROBOT MÓVIL MORLACO 165
A.1 Diseño 165
A.2 Sensores 166
A.3 Actuadores 167
A.4 Software de control 168

B ODOMETRÍA 171
B.1 Desplazamiento relativo 171
B.2 Desplazamiento global 172
B.3 Errores por aproximación para ángulos pequeños 174

C DETECCIÓN DE BORDES VERTICALES 175
C.1 Introducción 175
C.2 Gradiente de una imagen 176
C.3 Operadores de detección de bordes 177
C.4 Convolución separable 179

D LA IMAGEN INTEGRAL 181
D.1 Cálculo de la imagen integral 181
D.2 Sumas de píxeles en áreas rectangulares 182

E PROPAGACIÓN DE LA INCERTIDUMBRE 185
E.1 Propagación lineal 185
E.2 Propagación no lineal 185
E.3 Ley de propagación de la incertidumbre 187

F VISUALIZACIÓN DE LA INCERTIDUMBRE 189
F.1 Introducción 189
F.2 Incertidumbre en la posición del robot 189
F.3 Cálculo de la elipse de error 190
F.4 Resumen 194

BIBLIOGRAFÍA 197

ÍNDICE DE FIGURAS

Figura 1.1.1 Campos de investigación 4
Figura 1.1.2 Orgranización de la tesis 7
Figura 2.1.1 Mapa de la epidemia de cólera de 1854 22
Figura 2.1.2 Mapa de la campaña de Napoleón contra Rusia . . 22
Figura 2.1.3 Ejemplo de plano rasterizado 26
Figura 2.1.4 Ejemplo de planos vectoriales 26
Figura 2.1.5 *Geomery Object Model* 27
Figura 2.1.6 El concepto de "Geometría" 28
Figura 2.1.7 Evolución del procesamiento de datos espacial. . . 30
Figura 2.1.8 Ejemplos de relaciones topológicas 32
Figura 3.3.1 Sistemas de referencia 48
Figura 3.4.1 Transformación de coordenadas 53
Figura 3.4.2 Modelo de ruido para una puerta 54
Figura 4.1.1 Láser Sick Lms 200 63
Figura 4.1.2 Características de un barrido láser 63
Figura 4.1.3 Calibración de los errores del láser 65
Figura 4.1.4 Distancias promediadas para la calibración 65
Figura 4.1.5 Ilustración del siglo xvii de una cámara oscura . . 66
Figura 4.1.6 Webcam 9000 pro de Logitech 67
Figura 4.1.7 Modelo de lente delgada 68
Figura 4.1.8 Cámara estenopéica ideal 69
Figura 4.1.9 Modelo de cámara estenopéica 69
Figura 4.1.10 Proyección de un punto en la imagen 70
Figura 4.1.11 Encoder óptico rotatorio 70
Figura 4.2.1 Sistemas de referencia del robot, láser y cámara. . . 73
Figura 4.2.2 Rotación del láser y detección de un punto 74
Figura 4.2.3 Calibración de los parámetros del láser 75
Figura 4.2.4 Un punto en distintos sistemas de referencia 78
Figura 4.2.5 Proyección de un punto en una imagen 79
Figura 4.2.6 Características del sensor CMOS 79
Figura 4.2.7 Imagen para la calibración de la cámara 80
Figura 4.2.8 Datos láser correspondientes a la fig. 4.2.7 81
Figura 4.2.9 Ejemplo de fusión de datos 82
Figura 5.1.1 Esquema del sistema de localización propuesto . . 87
Figura 5.1.2 Proyección de datos láser en una imagen 90
Figura 5.1.3 Proyección de paredes en una imagen 91
Figura 5.1.4 Detección de líneas verticales 91
Figura 5.1.5 Resultado final de la detección de puertas. 92
Figura 5.1.6 Ciclo predicción-corrección del EKF 93
Figura 5.2.1 Ejemplo de barrido láser 95
Figura 5.2.2 Cálculo del vector $\vec{w}_i$ 97
Figura 5.2.3 Histograma angular para la fig. 5.2.1 98
Figura 5.2.4 Barrido láser rotado y alineado 99
Figura 5.2.5 Tramos de superficie en la pared izquierda 100

Figura 5.2.6 Histograma-X para la pared izquierda 101
Figura 5.2.7 Histograma-X para la pared derecha 101
Figura 5.2.8 Detección final de paredes con láser 102
Figura 5.3.1 Regiones de interés 104
Figura 5.3.2 Escalado y traslación de una *wavelet* de *Haar* 105
Figura 5.3.3 Tipos de *wavelet* de *Haar* en 2D y 3D 105
Figura 5.3.4 Cálculo de una característica pseudo-Haar vertical 106
Figura 5.3.5 Detección de líneas verticales 108
Figura 5.3.6 Resultado del algoritmo de ventanas deslizantes . 110
Figura 5.3.7 Superposición del resultado del algoritmo de ventanas deslizantes sobre la imagen de ejemplo . . . 110
Figura 5.3.8 Líneas verticales detectadas 111
Figura 5.3.9 AFD para detectar los laterales de una puerta 112
Figura 5.3.10 Puertas detectadas con marco doble 113
Figura 5.3.11 Puerta detectada con marco simple 114
Figura 5.4.1 Plano digital con información por capas 116
Figura 5.4.2 Capas existentes en un archivo DXF. 117
Figura 5.4.3 Opciones de exportación a formato shapefile. . . . 117
Figura 5.4.4 Esquema conceptual de PostGIS 118
Figura 5.4.5 Tabla `SPATIAL_REF_SYS` de PostGIS 119
Figura 5.4.6 Ejemplo de tabla `GEOMETRY_COLUMNS` 120
Figura 5.4.7 Tabla `segundaplanta` 120
Figura 5.4.8 Consulta de una puerta concreta 121
Figura 5.4.9 Consulta de las puertas dentro de una distancia . . 121
Figura 5.5.1 Localización mediante EKF 123
Figura 6.1.1 Distintos entornos de prueba 135
Figura 6.1.1 Distintos entornos de prueba 136
Figura 6.1.2 Anchura en píxeles 137
Figura 6.1.3 Ejemplos de detección de diferentes puertas 140
Figura 6.1.4 Falsos positivos 141
Figura 6.1.5 Interpolación de distancias. 141
Figura 6.2.1 Intercomunicación de los módulos desarrollados . 150
Figura 6.2.2 Ausencia de características en un pasillo 150
Figura 6.2.3 Misión en un pasillo de gran longitud 151
Figura 6.2.4 Trayectoria realizada durante la misión 1 152
Figura 6.2.5 Error estadístico en la misión 1 153
Figura 6.2.6 Área de la elipse de error en la misión 1 154
Figura 6.2.7 Misión en el interior de un departamento 155
Figura 6.2.8 Trayectoria realizada durante la misión 2 155
Figura 6.2.9 Error estadístico en la misión 2 156
Figura A.1 Robot *Morlaco* 166
Figura A.2 Sensores de *Morlaco* 166
Figura A.3 Controladora AX2850 168
Figura A.4 Módulos software desarrollados 169
Figura B.1 Desplazamiento local del robot 171
Figura B.2 Desplazamiento global del robot 173
Figura C.1 Detección de bordes manual 176
Figura D.1 Imagen integral y cálculo de un área 181
Figura D.2 Ejemplo práctico de cálculo de un área 183
Figura E.1 Sistema estocástico en una dimensión 185
Figura E.2 Propagación lineal del error 186
Figura E.3 Propagación no lineal del error 186
Figura E.4 Sistema estocástico multivariable 187
Figura F.1 Visualización de una bivariante gaussiana 191

Figura F.2 Isolíneas de probabilidad constante 191
Figura F.3 Parámetros de una elipse 192

ÍNDICE DE TABLAS

Tabla 2.1.1 División de tareas entre Sistema de Información Geográfica (SIG) y bases de datos espaciales. 30
Tabla 4.1.1 Parámetros variables del láser en función de la resolución angular. 64
Tabla 4.1.2 Especificaciones del láser 64
Tabla 4.1.3 Especificaciones de la cámara 67
Tabla 4.2.1 Parámetros resultantes de la medición del triángulo de calibración. 76
Tabla 4.2.2 Parámetros resultantes de la calibración del láser. . 76
Tabla 4.2.3 Parámetros intrínsecos de la cámara. 77
Tabla 4.2.4 Parámetros extrínsecos de la cámara. 77
Tabla 4.2.5 Coordenadas para la calibración de la cámara. . . . 82
Tabla 4.2.6 Parámetros resultantes de la calibración. 82
Tabla 5.3.1 Valores de las constantes utilizadas para calcular el valor de una ventana de Haar. 108
Tabla 5.4.1 Ejemplos de objetos geométricos en formato WKT. . 118
Tabla 6.1.1 Resultados de la clasificación de puertas 138
Tabla 6.1.2 Hardware de los entornos de prueba. 142
Tabla 6.1.3 Tiempos de ejecución del algoritmo de detección de puertas en diferentes entornos de prueba 143
Tabla 6.1.4 Tiempos de procesamiento de la imagen integral . 144
Tabla 6.1.5 Tiempos de procesamiento de la convolución. . . . 145
Tabla A.1 Peso de los componentes de *Morlaco* y características de sus motores 167

ÍNDICE DE ALGORITMOS

5.2.1 Cálculo de la dirección principal de un barrido 97
5.2.2 Detectar y calcular la distancia a la pared izquierda 102

5.3.1 Detección de líneas verticales 109
5.3.2 Detección de puertas con marco doble 113
5.3.3 Detección de puertas con marco simple 114

D.1 Cálculo de la imagen integral 182

ACRÓNIMOS

AFD Autómata Finito Determinista

AGV Automated Guided Vehicle

API Application Programming Interface

BNF Backus-Naur Form

BIM Building Information Modeling

CAD Computer-Aided Design

CAFM Computer-Aided Facility Management

CCD Charge-Coupled Device

CDL Configuration Description Language

CNL Configuration Network Language

CGIS Canada Geographic Information System

CLI Canada Land Inventory

CMOS Complementary Metal-Oxide-Semiconductor

CRT Cathode Ray Tube

CUDA Compute Unified Device Architecture

DWG AutoCAD Drawing

DXF AutoCAD Drawing Interchange File

EIF Extended Information Filter

EKF Extended Kalman Filter

EPSG European Petroleum Survey Group

GPU Graphics Processing Unit

GroUsal Grupo de Robótica de la Universidad de Salamanca

ISO International Organization for Standardization

IPC Inter Process Communication

IPT InterProcess communications Toolkit

IWMS Integrated Workplace Management System

KF Kalman Filter

OGC Open Geospatial Consortium

RDE Robotic Development Environment

RFID Radio Frequency Identification

RDI Region de Interés

SFA Simple Features Access

SGBDOR Sistema Gestor de Bases de Datos Objeto-Relacional

SGBDR Sistema Gestor de Bases de Datos Relacional

SIG Sistema de Información Geográfica

SLAM Simultaneous Localization and Mapping

SQL Structured Query Language

SDT Spatial Data Types

STIG Servicio Transfronterizo de Información Geográfica

UAV Unmanned Aerial Vehicles

UGV Unmanned Ground Vehicles

WKB Well-Known Binary

WKT Well-Known Text

Parte 1

INTRODUCCIÓN

CAPÍTULO 1.1

INTRODUCCIÓN

Hace cientos de miles de años, el hombre primitivo comenzó a utilizar madera y piedra para construir sus primeros refugios en un intento de protegerse de los depredadores y combatir las inclemencias del tiempo. Desde entonces, y a lo largo de la historia, se han desarrollado nuevos materiales de construcción y las edificaciones fabricadas por el ser humano han ido evolucionado con el fin de cubrir una amplia gama de necesidades: viviendas, aeropuertos, fábricas, hospitales, templos, centros comerciales, discotecas, hoteles, bibliotecas, centros educativos, etc. Esta diversificación en cuanto a la función del inmueble refleja el hecho de que los edificios se han convertido en nuestro hábitat principal, y es en ellos donde pasamos la mayor parte de nuestro tiempo. Por esta razón, existe un esfuerzo constante por modernizarlos e incluso dotarlos de automatización e inteligencia.

Son muchos los edificios que hoy en día utilizan microprocesadores, sensores y actuadores para medir y controlar la iluminación, la climatización, o los sistemas de seguridad de forma automatizada. Sin embargo, la tecnología actual es todavía bastante limitada en cuanto a inteligencia y autonomía, y los avances más innovadores en estos aspectos se están realizando gracias al progreso de la *computación ubicua* y la *inteligencia ambiental* [21].

El concepto de computación ubicua se refiere a un mundo en el que los computadores estén tan perfectamente integrados e interconectados entre sí en cualquier parte del entorno, que se puedan utilizar sin ser conscientes de estar interactuando con ellos [111]. Por ejemplo, en el interior del edificio, un entorno de computación ubicua podría incluir sensores biométricos en la ropa, sensores de temperatura en las paredes y sensores de presión en el suelo, todos ellos robustamente interconectados entre sí de modo que controlaran la iluminación y la climatización de las habitaciones continuamente y de manera imperceptible para el usuario. Otro ejemplo es el de frigoríficos capaces de saber qué alimentos contienen, si están en buen estado o no, capaces de ofrecer diferentes menús en función de los alimentos que contienen y comunicarse directamente con el mercado cuando se necesiten adquirir nuevos alimentos. La investigación en computación ubicua suele estar centrada en el desarrollo de redes de dispositivos y sensores [68].

La inteligencia ambiental, aunque comparte ciertos elementos con la computación ubicua, está más relacionada con el campo de la inteligencia artificial, haciendo énfasis en tareas de visión, procesamiento del lenguaje, abstracción del conocimiento y procesamiento espacio-temporal [30]. Para poder aplicar la inteligencia ambiental en el mundo real es imprescindible disponer de la capacidad de percibir y actuar en el entorno. Esto se puede lograr de dos maneras: a través de dispositivos inteligentes o bien por medio de robots. Por una parte, la utilización

de dispositivos inteligentes demuestra la relación existente entre la inteligencia ambiental y la computación ubicua. Por otra parte, si se utilizan robots para interactuar con el entorno, se están enlazando los campos de la inteligencia ambiental con la robótica autónoma, dando lugar a un campo más complejo y con mucho potencial por investigar. Aunque esquematizar la relación entre los diferentes campos citados no resulta trivial y se podría interpretar y visualizar de diferentes maneras, la representación de *Saffioti* y *Broxvall* [93] resulta bastante clarificadora, tal y como se muestra en la figura 1.1.1. Como se puede observar, la

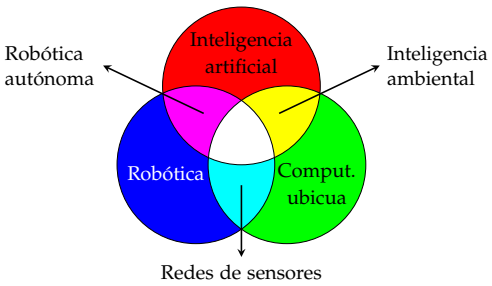


Figura 1.1.1: Relaciones entre diferentes campos de investigación.

integración de la robótica autónoma junto con la inteligencia ambiental correspondería a la convergencia de tres campos fundamentales para poder integrar un robot en un edificio: la robótica, la inteligencia artificial y la computación ubicua.

Sin lugar a dudas, uno de los pasos imprescindibles para lograr que los robots puedan estar presentes de forma natural en edificios inteligentes, es que adquieran habilidades similares a las humanas. Además de nuestra inteligencia, una de las capacidades más extraordinarias que el ser humano posee es la comprensión del espacio y la posibilidad de desarrollar cualquier tipo de tarea espacial con gran precisión. Esta capacidad es fruto de la permanente necesidad del ser humano de interactuar con el mundo que lo rodea, y se ha desarrollado y perfeccionado a lo largo de miles de años. Por lo tanto, uno de los requisitos necesarios en cualquier robot que se desee integrar en entornos humanos es que sea capaz de comprender, interpretar y representar el entorno de manera eficiente, consistente y lo que es más importante de manera compatible con el ser humano [108].

1.1.1 MOTIVACIÓN

El problema del conocimiento y la comprensión espacial ha dado lugar a numerosos desafíos. Quizás uno de los más importantes abordados hasta la fecha sea cómo conseguir que un robot se mantenga localizado mientras se desplaza a través de su entorno. Afortunadamente, durante las dos últimas décadas se han realizado importantes avances en el desarrollo de estrategias de localización [40], construcción de mapas [72, 104] e incluso la unión simultánea de ambas, más conocida como SLAM [10]. Como resultado, se han desarrollado diferentes métodos probabilísticos muy robustos, como por ejemplo técnicas de filtrado de Kalman [52, 69] filtros de partículas [106], o algoritmos de esperanza-maximización [71], entre otros. A pesar de esta evolución tan considerable, los métodos que ofrecen una mayor fiabilidad están basados en datos sensoriales en bruto [20, 105], o en características de

bajo nivel, como líneas [7]. Esto es un claro indicador de que, a pesar de que existen técnicas muy avanzadas para realizar tareas de localización y construcción de mapas, los robots son todavía incapaces de extraer información simbólica del entorno con la misma facilidad y precisión que lo hacemos los humanos.

Por lo tanto, existe una clara necesidad de aumentar el nivel de abstracción que puede manejar un robot, lo que está considerado como uno de los mayores desafíos actuales dentro del problema de la cognición espacial. En otras palabras, los robots deberían ser capaces de extraer y manejar información simbólica. Esta idea no es nueva, y es un problema estudiado desde los comienzos de la inteligencia artificial en la década de 1960. Es interesante destacar cómo las expectativas eran ingenuamente optimistas por aquel entonces, puesto que se pensaba que problemas como el reconocimiento de objetos podrían resolverse definitivamente mediante un proyecto de un par de meses de duración [87]. Sorprendentemente, cincuenta años más tarde, mientras que el cerebro humano sigue reconociendo cualquier tipo de objeto sin ninguna dificultad, todavía no se ha averiguado cómo resolver este problema computacional tan extremadamente complicado. La dificultad reside en que un mismo objeto puede dar lugar a la proyección de un número infinito de imágenes distintas en el plano de la imagen del observador, dependiendo de la posición, orientación, tamaño e iluminación del objeto [89].

Además de la estimación de la posición y el manejo de la información simbólica, existe otro problema relacionado con la cognición espacial que resulta muy interesante: cómo representar datos espaciales de forma escalable, interoperable y compatible con el ser humano. Probablemente, la única tecnología actual que permite manejar la información espacial de acuerdo a estos tres requisitos son los *Sistemas de Información Geográfica* (SIG) [65]. En primer lugar, los SIG tienen la capacidad de manejar a la perfección datos a cualquier escala, desde muebles en una habitación hasta datos de una ciudad, país o continente. En segundo lugar, la interoperabilidad de los SIG con otros estándares es cada vez mayor. Gracias a recientes esfuerzos de compatibilidad por parte de la industria, hoy en día se pueden utilizar junto con sistemas de gestión de infraestructuras asistidos por computador (CAFM), sistemas de gestión integrada del espacio de trabajo (IWMS), modelado de información de construcción (BIM) o diseño asistido por computador (CAD), entre otros [91]. La información espacial es difícil de obtener, y la interoperabilidad permite a los SIG reutilizar modelos espaciales de lugares previamente modelados. En tercer lugar, los SIG son completamente compatibles y fácilmente interpretables por el ser humano, porque en sus orígenes fueron diseñados con la intención de analizar y visualizar datos geoespaciales. Además la información se puede modelar simbólicamente y separar en capas independientes.

Al integrar un sistema de localización con un SIG, se abre la posibilidad por ejemplo, de mantener localizado al robot en cualquiera de los sistemas con los que un SIG sea compatible, como por ejemplo en sistemas de gestión de infraestructuras. De este modo se puede incluir al robot como un recurso más del sistema y facilitar su gestión.

1.1.2 OBJETIVOS

Tanto en sistemas de inteligencia ambiental como en sistemas de gestión de inmuebles, la necesidad de mantener perfectamente localizado a un robot autónomo en el interior de un edificio donde desarrolle su servicio se torna imprescindible, sin importar que la edificación sea un hospital, un centro educativo, una nave industrial u otro tipo de inmueble. En el caso de un sistema de inteligencia ambiental el robot actúa como un recurso que permite percibir y actuar en el entorno, mientras que en el caso de un sistema de gestión de inmuebles, el robot sería un recurso más al igual que los recursos humanos o los recursos espaciales, por ejemplo.

Así pues, el principal objetivo de esta tesis es afrontar el problema de la localización de un robot móvil en entornos interiores. A pesar de que este problema ha sido abordado con anterioridad, los métodos más eficaces utilizan datos sensoriales en bruto en lugar de extraer de información simbólica del entorno. Por lo tanto, desarrollar un sistema de localización basado en la utilización de información simbólica permitiría avanzar en el problema del conocimiento espacial.

A partir de esta idea, los objetivos concretos de esta tesis son los siguientes:

- Desarrollar un sistema de extracción de información simbólica del entorno que permita detectar puertas y conocer su ubicación en el mundo con respecto al robot, todo ello en tiempo real. Para resolver este problema proponemos fusionar datos láser y visión monocular, construyendo un sensor virtual con mayor capacidad que la visión o el láser de manera independiente.
- Integrar a nuestro robot en un *Sistema de Información Geográfica* (SIG). Gracias a esto el robot podrá realizar consultas espaciales al mapa almacenado en la base de datos espacial subyacente, así como acceder a la información simbólica del entorno (puertas, ventanas, habitaciones, etc.). Además la información se almacena de forma compatible con el ser humano y gracias a la interoperabilidad de los SIG se abre la posibilidad de utilizar al robot como un recurso más en sistemas de gestión de inmuebles.
- Diseñar y desarrollar un sistema de localización basado en el filtro extendido de Kalman (EKF), e integrarlo junto con nuestro sistema de detección de puertas y el SIG. De este modo el robot se mantendrá localizado utilizando la información simbólica que extrae del entorno, contrastándola con los datos espaciales del mapa almacenados en el SIG. El sistema de localización debe funcionar de manera totalmente autónoma, sin intervención humana.

1.1.3 ORGANIZACIÓN DE LA TESIS

La tesis se ha dividido en siete partes, donde cada parte está formada por un conjunto de capítulos. Para ilustrar mejor la organización de las partes y su relación con los objetivos, se ha creado el organigrama de la figura 1.1.2. Cada uno de los objetivos está marcado con un color diferente, y además de las partes, también se muestra la relación de los apéndices (A-F) con las partes donde resultan relevantes.

En primer lugar es importante destacar que el objetivo global de la tesis es desarrollar un sistema de localización, de modo que este

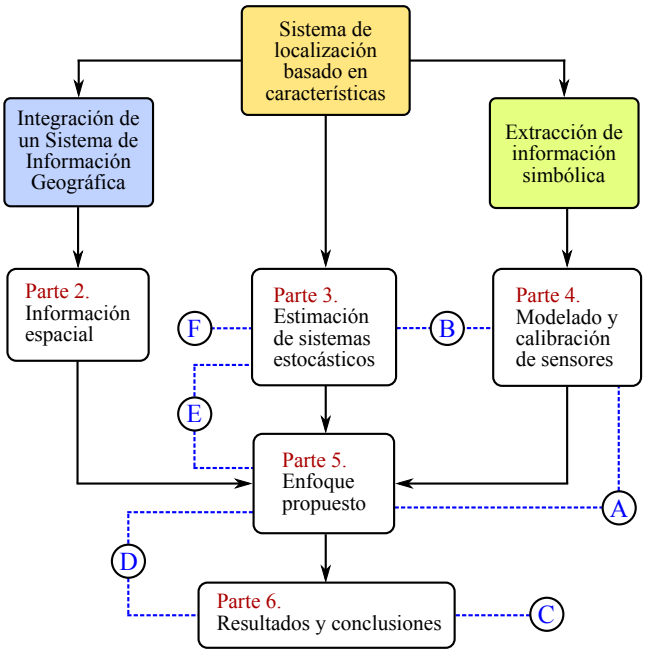


Figura 1.1.2: Organigrama que relaciona los objetivos propuestos con las partes de la tesis y los apéndices.

objetivo engloba a los otros dos. La parte *Introducción* no aparece en el organigrama para no complicarlo, pero consiste en el capítulo actual y una revisión del estado del arte relacionada con los tres objetivos de la tesis. El resto de partes se describen a continuación.

Parte 2 – *Información espacial*: analiza desde un punto de vista teórico los Sistemas de Información Geográfica, sus características y aplicaciones, su capacidad para manejar datos espaciales, y cómo han evolucionado hasta utilizar bases de datos espaciales para almacenar la información.

Parte 3 – *Estimación de sistemas estocásticos*: incluye dos capítulos teóricos donde se describe tanto el filtro de Kalman (para sistemas lineales) como el filtro de Kalman extendido (para sistemas no lineales). Los dos siguientes capítulos ponen esta teoría en práctica e introducen el modelo del sistema aplicado a nuestro robot, así como el modelo de observación de información simbólica del entorno, que son fundamentales para desarrollar e implementar el sistema de localización.

Parte 4 – *Modelado y calibración de sensores*: en el primer capítulo se estudian y modelan los sensores a bordo del robot (láser, cámara y *encoders*). En el segundo, se describe la utilización de un procedimiento de calibración para fusionar la información sensorial a bajo nivel entre el láser y la cámara. Esta parte resulta fundamental para comprender el método de extracción de características del entorno que se propone en la parte 5.

Parte 5 – *Enfoque propuesto*: aquí se describe en primer lugar la perspectiva general del sistema propuesto para resolver los tres objetivos planteados. Los dos capítulos siguientes detallan la extracción de características mediante la fusión sensorial de láser y visión. A continuación se explica cómo almacenar y consultar información espacial en un sistema de información geográfica. El último capítulo de esta parte describe la utilización de un filtro extendido de Kalman para resolver

el problema de localización mediante extracción de características y sistemas de información geográfica.

Parte 6 – *Resultados y conclusiones*: presenta los resultados del sistema de extracción de características junto con los resultados de las pruebas de localización en diferentes casos experimentales que verifican el correcto funcionamiento de nuestra propuesta. El capítulo de conclusiones analiza los logros realizados y nuevas posibilidades de trabajo futuro.

Parte 7 – *Apéndices*: El apéndice A explica el diseño y el sistema de control del robot utilizado en este trabajo, mientras que en el apéndice B se deriva matemáticamente el modelo odométrico del robot, que resulta imprescindible para el sistema de localización propuesto. El apéndice C presenta la subjetividad existente en el problema de detección de bordes y cómo resolverlo mediante filtros de convolución, que se utilizarán como comparativa en la parte de resultados. La imagen integral se describe en el apéndice D, y es una técnica clave para poder lograr la extracción de características en tiempo real. Finalmente los dos últimos apéndices, E y F, se refieren a la propagación y la visualización de la incertidumbre, respectivamente.

CAPÍTULO 1.2

ANTECEDENTES Y ESTADO DEL ARTE

En este capítulo se ofrece una pequeña introducción al problema de la localización, así como de la evolución que han ido sufriendo las diferentes técnicas de localización desarrolladas a lo largo de la historia. A continuación se revisan los trabajos más destacados relacionados con los objetivos principales de la tesis, comenzando por el uso de los sistemas de información geográfica en robótica. En segundo lugar se estudian los sistemas de extracción de características relacionados con la detección de puertas, que es el objeto que se ha decidido emplear como elemento clave para el sistema de localización que proponemos. Finalmente, se estudia el uso de líneas verticales para la localización, dado que nuestro sistema de detección de puertas se basa en la detección de líneas verticales.

1.2.1 LOCALIZACIÓN

La capacidad de un robot móvil para determinar su ubicación en el espacio es una tarea imprescindible para poder navegar de forma totalmente autónoma [9]. El conocimiento de la propia posición, así como de otros objetos o características de interés en el entorno del robot, son los cimientos básicos sobre los que se sustentan las operaciones de navegación de más alto nivel. Por ejemplo, la localización permite que sea posible planificar trayectorias para llegar a un destino concreto mientras se evitan obstáculos, y es fundamental para tareas más complejas como construir mapas de entornos desconocidos. Sin la capacidad de localización, los robots estarían condenados a interactuar con el entorno a través de comportamientos reactivos, y serían incapaces de planificar acciones más allá de su campo de percepción local.

1.2.1.1 *Perspectiva general de la localización*

Se puede considerar que el origen de la localización de robots móviles se remonta a 1954 [110], cuando se diseñó el primer vehículo de guiado autónomo o AGV (*Automated Guided Vehicle*), consistente en un vehículo de remolque de un almacén de provisiones, modificado seguir una línea de tensión del almacén de la que obtenía la energía necesaria para funcionar. A finales de la década de 1970, gracias al desarrollo de los circuitos integrados, los AGV eran controlados por ordenador y podían seguir trayectos por inducción magnética a través de cables enterrados bajo la superficie o bien por medio de rayas visualmente identificables en el suelo [35]. El inconveniente fundamental de estas técnicas es que la navegación se limita a recorridos fijos, y por lo tanto el uso de los AGV queda restringido a tareas de carácter repetitivo, de ahí que sean soluciones ampliamente utilizadas en entornos industriales. El

seguimiento de este tipo de trayectorias predefinidas fue un precursor de la localización autónoma al precipitar la necesidad de estrategias de localización más flexibles [9].

Más adelante se introdujo una mayor flexibilidad en la navegación mediante técnicas de estimación de la posición utilizando balizas artificiales. Este tipo de sistemas incluyen balizas activas de infrarrojos o ultrasonidos, o balizas pasivas como marcadores reflectantes [16], y permiten a los robots móviles localizarse con respecto a la ubicación de las balizas. Esto supuso un importante avance en las estrategias de localización, dado que las trayectorias de los robots podían redefinirse por software sin la necesidad de modificar físicamente el entorno, abriéndose además las puertas para la generación de trayectorias adaptativas con capacidad para evitar obstáculos. No obstante, al igual que ocurre con los AGV guiados por cables o marcas en el suelo, esta técnica necesita alterar el entorno, puesto que depende de la instalación de las balizas en el entorno.

El siguiente avance en el desarrollo de las estrategias de localización fue la detección de características en el entorno, utilizándolas como puntos de referencia para estimar la posición del robot. Se entiende por *característica* un determinado objeto que el robot pueda reconocer mediante su sistema sensorial, como por ejemplo patrones gráficos introducidos artificialmente, o bien elementos ya existentes previamente en el entorno. En el primer caso, se habla de características *artificiales*, que normalmente están diseñadas para ser detectadas fácilmente mediante visión artificial, como rombos [42] o círculos [53]. El segundo caso se refiere a lo que se conoce como características *naturales*, siendo las líneas verticales ampliamente utilizadas [58, 25, 78]. Tienen la ventaja de que eliminan la necesidad de modificar físicamente el entorno del robot, pero en cambio son mucho más difíciles de detectar que las características artificiales. Para determinar la posición del robot mediante esta técnica, se necesita disponer *a priori* de un mapa del entorno, para comparar las observaciones con los datos del mapa. La detección de características naturales dio lugar al problema de la *asociación de datos* [31]: encontrar la correspondencia entre los elementos de dos conjuntos de datos. A pesar de que este problema también existía en la localización mediante balizas artificiales, se podía eliminar muy fácilmente haciendo que cada baliza fuera identificable de manera única.

1.2.1.2 *Técnicas de localización*

Una de las técnicas más básicas de localización es la que se conoce como *dead reckoning* , y que consiste simplemente en calcular la posición del robot a partir de sensores propioceptivos, como acelerómetros o *encoders*. El principal problema de este método es que cada nueva estimación de la posición incluye un pequeño error, y estos errores se acumulan como parte del proceso de cálculo, de modo que la incertidumbre en la estimación de la posición se incrementa continuamente en el tiempo. Mejorar la precisión del modelo matemático del sistema únicamente serviría para retrasar, pero no eliminar, el incremento en la incertidumbre. Después de un periodo de tiempo lo suficientemente grande, la estimación de la posición sería tan incierta que no serviría absolutamente para nada. No obstante, a pesar de que el *dead reckoning* a largo plazo es un método de localización insuficiente, es un mecanis-

El término 'dead reckoning' data del siglo XVII *y se refería al proceso de estimar la posición de un barco a partir de la velocidad, el rumbo y el tiempo transcurrido desde la última posición conocida, sin tener en cuenta observaciones externas (astros, normalmente) que pudieran ayudar a mejorar la posición.*

mo muy útil a corto plazo si se combina con observaciones externas que corrijan la estimación.

Para evitar que la incertidumbre en la posición aumente constantemente, no se puede contar únicamente con medidas incrementales como las que proporcionan los sensores propioceptivos, sino que se deben incorporar medidas absolutas o exteroceptivas. Un mapa del entorno en el que se encuentre definida la posición de objetos que el robot sea capaz de detectar proporciona las medidas exteroceptivas necesarias. Así, comparando las características detectadas con las características presentes en el mapa, el robot puede determinar su posición tanto de manera relativa como de manera absoluta.

Cuando se utilizan características para la localización, se suele utilizar algún tipo de mapa donde se encuentran almacenadas las características observables en coordenadas globales. Las observaciones que efectúa el robot vienen dadas en el marco de referencia local del robot. El procedimiento de localización consiste en determinar la transformación de coordenadas que hace coincidir las medidas locales con las absolutas.

Existen algunos sensores como las radio-balizas, lectores de códigos de barras o de patrones especializados, que siempre que el conjunto de características sea distinguible unas de otras, al detectar la característica se podrá identificar exactamente en qué lugar se está. Sin embargo, en general los datos en bruto de los sensores no contienen ninguna relación entre los datos y su localización en el mapa, sino que es el robot el que debe utilizar algún sistema de localización para determinar su ubicación.

1.2.2 LOS SIG EN ROBÓTICA

Desde hace algo más de una década se comenzó a explorar el uso de los sistemas de información geográfica en tareas de robótica móvil, y poco a poco se ha ido incrementando su integración con los robots, siendo cada vez más habitual su utilización en este ámbito en los últimos años.

Uno de los primeros usos de los SIG en el campo de la robótica móvil es el diseño de una silla de ruedas robótica destinada para guiar a personas con discapacidad visual en entornos exteriores [77], donde el SIG se utiliza como la base del sistema de navegación y como modelo de carreteras, caminos, y marcas reconocibles. Los autores comparan al robot con a un perro guía, y concluyen que a pesar de que el robot tiene menor capacidad de movilidad, ofrece prestaciones adicionales de navegación, gracias al uso de la información contextual proporcionada por el SIG.

Los SIG en robótica son utilizados comúnmente para tareas de navegación en entornos exteriores. Los sensores de un robot le sirven para percibir su entorno más inmediato y extraer características, que pueden contrastarse a continuación con la información almacenada en el SIG. Por ejemplo, Bonnifait *et al.* [15] utilizan un escáner láser montado en un vehículo para detectar los bordes de la calzada en entornos urbanos y los comparan con una red de carreteras almacenada en una capa de un SIG, determinando así la posición del vehículo en todo momento. Cappelle *et al.* [23] disponen de un GIS en 3D donde está almacenado un modelo de ciudad de gran precisión, y utilizan un GPS para efectuar una estimación inicial de la posición del vehículo, que después complementan mediante cámara de visión y láser a través de un EKF. La información almacenada en un SIG también puede ser utilizada

simultáneamente por varios robots. Un ejemplo de ello es el trabajo de Mirats Tur *et al.* [73], que a partir de un SIG donde almacenan la información espacial de un entorno urbano, construyen mapas parciales en formato textual (comprensible por el hombre) y se los proporcionan a cada robot de un equipo multirobot a medida que los solicitan.

Un uso alternativo de los SIG es utilizarlos como un medio para compartir información geoespacial, en lugar de utilizarlos exclusivamente como una fuente de datos. Yoshida *et al.* [115], utilizan un equipo multirobot teleoperado que actúa durante un simulacro de rescate en un centro comercial ante una catástrofe debida a armas químicas, biológicas o nucleares. Los robots proporcionan simultáneamente información parcial del entorno que perciben, y se comunican entre ellos a través de DaRuMa [85], un prototipo de SIG desarrollado por el mismo equipo de investigación, diseñado para almacenar e integrar los datos 3D del entorno, la ubicación de las víctimas y las trayectorias de cada robot. Esta información puede ser utilizada posteriormente por otros robots o por equipos humanos de rescate.

Otra posible aplicación es en el ámbito militar, como ayuda para los vehículos aéreos no tripulados (UAV) y vehículos terrestres no tripulados (UGV). Los UAV en ocasiones pueden llegar a perder la comunicación con el operador humano que lo controla, y en estos casos es crucial que ellos mismos sean capaces de encontrar zonas seguras en las que realizar aterrizajes de emergencia de forma autónoma. Para afrontar este problema, Rackliffe *et al.* [90] combinan datos de diferentes SIG en busca de áreas pobladas y zonas posiblemente habitadas, como carreteras, aparcamientos, escuelas, hospitales o centros comerciales. A partir de estos datos crean mapas de probabilidad, donde el espacio queda clasificado en función de su seguridad para ser una zona apta para aterrizajes de emergencia. Estos mapas también los emplean, en el caso de los UGV, como información contextual del entorno para dotarlos de una mayor autonomía en las misiones tácticas, de modo que sin intervención humana puedan permanecer lejos de zonas habitadas por personal civil.

Uno de los grandes beneficios de incorporar los sistemas de información geográfica a la robótica móvil es que estos sistemas son bastante compatibles con el hombre. Cualquier sistema robótico actual necesita interactuar con el ser humano en una o varias etapas de su ciclo de vida, y los datos almacenados en los SIG facilitan esta comunicación hombre-robot, como destacan en [73].

1.2.3 DETECCIÓN DE PUERTAS

En el interior de un edificio, se pueden extraer diferentes características de alto nivel que sirven como fuente de información simbólica. De entre los diferentes objetos existentes, las puertas son uno de las más frecuentes y más regularmente repartidos en los edificios, y por ello se han elegido como la base del sistema de localización de nuestro robot móvil.

A la hora de afrontar el problema de la detección de puertas, quizás uno de los planteamientos más simples sea clasificar las puertas en función de su anchura. En [12, 96] utilizan un escáner láser 2D para realizar esta medida. De manera similar, Cariñena *et al.* [24] emplean un sistema basado en lógica difusa y sensores de ultrasonidos, con la principal limitación de que no pueden detectar más de una puerta

simultáneamente, y ésta tiene que estar como máximo a un metro de distancia del robot. Anguelov *et al.* [4] utilizan un láser 2D combinado con una franja de color extraída de una cámara omnidireccional. Su principal fuente de detección de puertas la constituyen los datos del láser, basándose en la suposición de que las puertas son aquellos objetos dinámicos del entorno que cambian su orientación con el tiempo. Gracias a las tiras de color logran mejorar ligeramente su tasa de aciertos, suponiendo que se pueden definir todas las puertas mediante un modelo de color uniforme.

El principal inconveniente de las técnicas basadas en medidas de distancia es que no funcionan a menos que una puerta esté abierta o que el plano de la puerta sea claramente distinguible del plano de la pared. Cuando el plano de la puerta está alineado con el plano de la pared, lo cual es bastante habitual en algunos entornos, los sensores de distancia resultan completamente inútiles, mientras que un sensor de visión se tornaría imprescindible. El principal problema de la visión monocular es que no permite detectar distancias. No obstante, cabe la posibilidad de combinar dos o tres cámaras dando lugar a sistemas de visión estéreo (binoculares o trinoculares, respectivamente), aunque el coste de procesamiento de la imagen se multiplica. Por ejemplo, Kim y Nevatia [55], utilizan un sistema de visión trinocular para extraer bordes verticales y horizontales, utilizando distancias reales (disponibles gracias a la visión estéreo) entre los bordes detectados para decidir qué clasifican como puerta.

A pesar de la innegable utilidad de las medidas de distancia reales (disponibles mediante visión estéreo, láseres o sónares), numerosos investigadores han utilizado únicamente la visión monocular como su única fuente de información. Esta decisión les obliga a idear métricas de distancia basadas en píxeles para poder clasificar como puertas los objetos que detecten en la imagen. Un importante inconveniente que afecta a esta técnica es que el éxito de los clasificadores se ve restringido únicamente a aquellas puertas que estén a una determinada distancia y orientación de la cámara del robot. Normalmente, sólo logran reconocer las puertas cercanas y paralelas al plano de la imagen. Si las puertas se encuentran perpendiculares y a media o gran distancia del plano de la imagen, sus proporciones se ven severamente afectadas por los efectos de la perspectiva. Por ejemplo, utilizando estimaciones de distancia basadas en píxeles, Monasterio *et al.* [74] miden la distancia entre bordes verticales detectados con un filtro de Sobel. Igualmente, pero con un sistema de lógica difusa, Muñoz-Salinas *et al.* [79] detectan características verticales y horizontales extraídas con la transformada de Hough. Yang y Tian [113] combinan la detección de bordes y esquinas con una serie de umbrales de distancia basados en píxeles. Murillo *et al.* [80] presentan un enfoque probabilístico basado en un modelo que contiene la forma geométrica y el aspecto de las puertas. El modelo de apariencia es aprendido a partir de imágenes etiquetadas a mano, y por lo tanto depende de los colores que tengan las puertas en las imágenes del conjunto de entrenamiento. El modelo geométrico se define mediante distancias en píxeles, de modo que, al igual que los métodos anteriores, el sistema falla cuando las puertas son perpendiculares y están lejos del plano de la imagen, porque en este caso únicamente tienen unos pocos píxeles de anchura. Además, los autores admiten tener una tasa relativamente alta de falsos positivos. Shi y Samarabandu [97] detectan puertas más lejanas mediante un análisis más complejo de la imagen,

que comprueba si los marcos de las puertas llegan hasta el suelo o no. A pesar de esta mejora, sorprendentemente el sistema no puede detectar puertas demasiado cercanas, porque se rechazan las puertas que no tengan visible el marco superior, o cuya anchura sea superior a un tercio de la imagen.

Una alternativa muy interesante a los métodos anteriores es el enfoque basado en aprendizaje automático *(machine learning)* de Chen *et al.* [27], donde obtienen un clasificador fuerte combinando varios clasificadores débiles gracias al uso del algoritmo Adaboost. Hensler *et al.* [51] utilizan la idea de Chen pero añadiendo datos de un láser 2D para estimar la concavidad y la anchura de las puertas. Entre algunos de los clasificadores débiles usados en estos dos trabajos se incluyen detectores de líneas verticales, de concavidad, comprobación visual de si hay un hueco debajo de la puerta candidata, color, textura, anchura de la puerta o comprobación de si existe una manilla en la puerta. El punto fuerte de estos métodos es que consiguen unas tasas muy bajas de falsos positivos. Sin embargo, se centran en puertas cercanas al sensor, y el tiempo de clasificación es elevado incluso con imágenes de baja resolución (320x240, 5fps), debido al gran número de clasificadores que se necesitan comprobar en cada imagen. En [28], Chen *et al.* mejoran sus resultados previos, utilizando un proceso de MonteCarlo vía cadenas de Markov *(Markov Chain MonteCarlo)*.

Finalmente es importante comentar que uno de los elementos más importantes que distingue a las puertas de otros objetos son las manillas. No obstante, son bastante difíciles de detectar debido a su tamaño extremadamente pequeño, y a que su forma es muy variable dependiendo de la puerta en concreto. Así, son detectados con mayor fiabilidad a distancias muy próximas. Aprovechando el hecho de que las manillas sobresalen de la superficie de las puertas, Rusu *et al.* [92] utilizan datos de distancia en 3D para girar el pomo o la manilla de la puerta y abrirla con el brazo robótico de un robot PR2. La capacidad de poder abrir puertas es muy interesante en cualquier robot de interiores, porque le permitiría mejorar su capacidad autónoma de navegación.

1.2.4 LÍNEAS VERTICALES

Los cambios de intensidad verticales en una imagen —también llamados bordes o características verticales— no es una idea nueva en la comunidad robótica, dado que desde los comienzos de la visión artificial se han empleado líneas verticales para resolver el problema de localización. Basándose en este tipo de características visuales, Sugihara [102] presentó una de las primeras técnicas para resolver el problema de la localización robótica mediante visión monocular. Suponiendo que los parámetros de calibración de la cámara eran conocidos, así como que las observaciones estaban libres de error, el método de Sugihara consistía en triangular la posición y orientación del robot utilizando un conjunto de bordes verticales extraídos en cada imagen. Para realizar la triangulación, calculaba el ángulo entre pares consecutivos de las características verticales detectadas, y buscaba la combinación más parecida en un mapa *a priori* de bordes verticales del entorno. Triangular implica que se necesita disponer simultáneamente de un mínimo de tres características diferentes para que la estimación de la posición y orientación no sea redundante. Mejorando este enfoque, Krotkov [60] propone un modelo similar con la diferencia de que tiene en cuenta

la incertidumbre en la posición de las características detectadas, usando un análisis del peor escenario. Atiya y Hager [8], bajo las mismas suposiciones, representan los errores usando tolerancias, lo que les lleva a plantear un algoritmo basado en conjuntos para resolver el problema de la asociación de datos. Cabe destacar que estos autores, a diferencia de los dos anteriores, tienen en cuenta en su planteamiento la posibilidad de observar falsos positivos, dado que es un problema que aparece frecuentemente en robótica móvil. Madsen y Andersen [67] demostraron que en el método de triangulación, cuando se usan grupos de tres características, en ocasiones se puede llegar a producir un error de tal magnitud que no se podría utilizar para problemas de navegación. No obstante, este inconveniente se puede subsanar prediciendo la incertidumbre de los grupos de tres características, y seleccionando los que tienen menor error. Todos estos métodos abordan el problema de la localización absoluta.

Otros modelos de error más sofisticados, tales como los que se emplean en los filtros de Kalman extendidos (EKF), también han sido aplicados a la localización basada en características verticales. Los métodos basados en el EKF normalmente proporcionan localización incremental. Una de las técnicas más populares consiste en leer de un mapa la ubicación de los bordes verticales, para proyectarlos en la imagen obtenida por la cámara, utilizando para ello la estimación de la posición del robot calculada en función de la odometría solamente. Esto permite reducir el espacio de búsqueda en la imagen a la hora de buscar cambios de intensidad verticales. Siguiendo este enfoque, Chenavier y Crowley [33] utilizan un EKF para fusionar odometría con visión, empleando la transformada de Hough para detectar bordes verticales. Las pruebas experimentales que realizan no entrañan gran dificultad, dado que utilizan un mapa con cinco características, las rutas que sigue el robot están preplanificadas y posicionan manualmente la cámara en la dirección de las características que esperan encontrar. Un trabajo similar en el enfoque pero mucho más minucioso y detallado es el sistema FINALE de Kosaka y Kak [58], que también confía en la transformada de Hough para la detección de líneas verticales, pero supone que existen modelos geométricos detallados en 3D del entorno. Arras y Tomatis [6, 7] también utilizan un EKF para fusionar características verticales en imágenes monoculares con paredes detectadas con un escáner láser 2D.

Otra serie de métodos probabilísticos utilizan el *Modelo de Simetrías y Perturbaciones* (SPmodel) de Tardós [103] para representar la incertidumbre. Algunos ejemplos del uso del SPmodel en conjunto con la detección de bordes verticales son la fusión de información sensorial de láser 2D y visión utilizando un EKF [81] o un Filtro de Información Extendido (EIF) [25], o la fusión de datos e intensidad procedentes de un láser 3D [83].

El problema de la localización de un robot móvil es una materia muy extensa, y hasta la fecha se han desarrollado diferentes técnicas que han funcionado con éxito. Por ejemplo, Bonin-Font *et al.* [14], o Desouza y Kak [34] constituyen amplias revisiones de trabajos relacionados con la visión empleada para la localización y la navegación, mientras que Filliat *et at.* [40] o Gutmann *et al.* [50] revisan el problema de localización utilizando distintos tipos de sensores. Sin embargo, y a pesar de que se ha dedicado un importante esfuerzo de investigación para resolver este problema tan esencial para la robótica móvil, los

robots inteligentes todavía presentan una carencia tremendamente importante: *la capacidad de comprensión espacial*, o en otras palabras, la habilidad para entender y comprender el mundo que nos rodea, así como la inmensa y diversa cantidad de objetos existentes. Se podría decir que esta falta de comprensión espacial es el talón de Aquiles de la robótica móvil actual, y su origen radica en que intentar reproducir el enorme conocimiento que los seres humanos poseemos acerca de nuestro entorno es una tarea tremendamente compleja [104].

CAPÍTULO 1.3

RESUMEN

Esta parte comienza con un capítulo de introducción al trabajo realizado, donde se describe la motivación de esta tesis, los objetivos que se han marcado para su desarrollo, así como la organización e interrelación de las diferentes partes, capítulos y apéndices.

El motivo principal que da lugar a este trabajo de investigación es avanzar en la integración de los robots en el habitat principal del ser humano: los edificios, o entornos interiores. Para lograr este avance, uno de los objetivos que se plantean es desarrollar un sistema de extracción de información simbólica del entorno, cuya principal contribución es en el problema del conocimiento espacial, una de las principales debilidades de la robótica autónoma hoy en día. Otro de los objetivos es integrar a nuestro robot en un Sistema de Información Geográfica, de modo que pueda aprovechar sus potentes características de procesamiento espacial. En este sentido, se contribuye además en la compatibilidad de los robots autónomos con el ser humano, gracias a la facilidad de uso y capacidad de visualización espacial de los SIG. Estos dos objetivos son la base del objetivo global de la tesis: desarrollar un sistema de localización para interiores basado en la extracción de información simbólica y en sistemas de información geográfica. Mantener a un robot localizado en un edificio resulta además fundamental para los sistemas en los que los robots constituyen un recurso, tales como los sistemas de inteligencia ambiental, o los sistemas de gestión de infraestructuras.

En el segundo capítulo de esta parte se presenta una introducción al problema de la localización y cómo ha evolucionado en el tiempo. Además, se revisan los trabajos más destacados relacionados con los objetivos de la tesis. En primer lugar, se estudia el uso de los sistemas de información geográfica en robótica. A continuación, dado que se ha optado por emplear puertas como características de alto nivel para el sistema de localización que proponemos, se comentan los principales detectores de puertas existentes en la actualidad. Para finalizar, teniendo en cuenta que la clave de nuestro detector de puertas son las líneas verticales, se incluye una revisión del uso de las líneas verticales en el problema de localización.

Parte 2

INFORMACIÓN ESPACIAL

CAPÍTULO 2.1

ALMACENAMIENTO Y ANÁLISIS DE INFORMACIÓN ESPACIAL

Las primeras pruebas del interés del ser humano en representar simultáneamente información gráfica y geográfica datan del paleolítico superior, hace aproximadamente unos 35000 años, en las cuevas de *Lascaux* (Francia). Las paredes de estas cuevas contienen pinturas rupestres de animales de la época, junto con líneas que se piensa que podrían representar rutas migratorias de las presas que cazaban [95]. Estas pinturas se podrían considerar como el primer intento humano de Sistema de Información Geográfica (SIG), puesto que contienen imágenes geográficas enlazadas con atributos semánticos, y probablemente surgieron tan pronto como el ser humano comenzó a tener una conciencia espacial de su entorno. Hoy en día se llevan a cabo estudios similares —utilizando SIG modernos— de rutas de migración de animales como el caribú o el oso polar mediante collares GPS, obteniendo inmediatamente información georeferenciada que permite diseñar programas de protección de estos animales y su hábitat.

Siglos más adelante, tras la llegada del Renacimiento a Europa, la cultura comienza a expandirse fuera del entorno eclesiástico y se producen avances en campos como la planificación urbana, la agrimensura, o las tácticas militares, que introducen nuevas necesidades de representación geográfica junto con una mayor información semántica. En la segunda mitad del siglo XIX aparecen las primeras aplicaciones del análisis de información espacial.

Un ejemplo fundamental se dio en 1854, cuando el médico inglés *John Snow* hizo algo revolucionario: convirtió datos geográficos en información geográfica y consiguió detener una terrible epidemia de cólera en el centro de Londres, donde se registraron más de 500 muertes en tan sólo 10 días [43]. En esa época, no se conocía cómo se transmitían las enfermedades, dado que estaba arraigada la teoría de los miasmas. El doctor *Snow* —escéptico de esta teoría— comenzó a registrar las muertes por cólera en un mapa de Londres, de manera muy similar a como se haría en un SIG actual, puesto que se convirtió en un mapa de información geográfica muy preciso. Pronto resultó evidente que el origen del brote epidémico estaba situado en una bomba de agua situada en *Broad Street* (fig. 2.1.1). A raíz de este descubrimiento, se clausuró la bomba de agua, y se logró contener la epidemia. Sin la innovadora idea de *Snow*, cientos de londinenses más habrían fallecido.

Según la teoría miasmática de la enfermedad, el cólera o la peste se transmitían a través de miasmas: efluvios malignos que desprendían cuerpos enfermos, materias corruptas o aguas estancadas.

Otro importante ejemplo coetáneo se da en 1869, cuando el ingeniero civil francés *Charles Joseph Minard* trazó su famoso mapa *Carte figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813* (fig. 2.1.2) ante la necesidad de representar información semántica compleja. En él, *Minard* documenta el intento de Napoleón de conquistar Rusia, haciendo énfasis en los devastadores

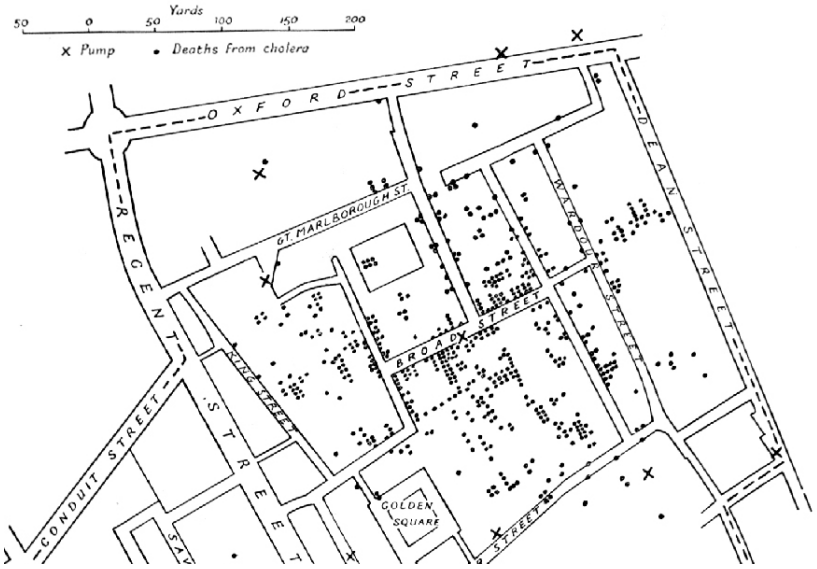


Figura 2.1.1: Fragmento del mapa original dibujado por el Dr. John Snow, donde se muestran mediante puntos las muertes por cólera durante la epidemia de Londres de 1854. Las cruces representan las bombas de agua.

efectos del clima invernal que afectó al ejército francés durante el transcurso de la campaña. En el mapa original, el número de hombres está representado por el ancho de las zonas coloreadas a razón de un milímetro por cada diez mil hombres; además están escritos en números en cada zona. El marrón designa los hombres que entran en Rusia, el negro, aquellos que la dejan. Incluye también una escala de distancia (en leguas francesas) y la porción inferior del gráfico muestra la temperatura ambiental durante el regreso del ejército desde Rusia (en grados en la escala de Réaumur). De este modo, *Minard* consigue integrar 5 elementos semánticos diferentes en una sola imagen en dos dimensiones: tamaño del ejército, dirección, ubicación geográfica, temperatura y fecha. Los SIG actuales necesitarían varias capas para lograr algo similar.

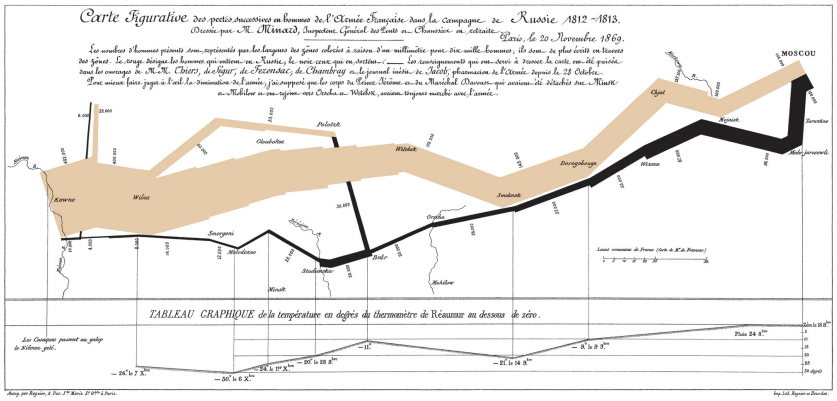


Figura 2.1.2: Mapa de Charles J. Minard (1869) que muestra el movimiento, las pérdidas humanas y la temperatura ambiental durante la campaña de Napoleón contra Rusia en 1812-1813.

Más recientemente, en la década de 1960, tuvo lugar el desarrollo de los Sistemas de Información Geográfica modernos. En 1958, el ministerio de ingeniería forestal y desarrollo rural de Canadá inició el proyecto *Canada Land Inventory* (CLI) para analizar sus tierras, determinando a lo largo del país si el suelo era apto para agricultura, bosques, u otros usos, con el fin de aprovechar mejor su utilización. Este proyecto generó una ingente cantidad de datos, y simultáneamente se emprendía otro proyecto, denominado *Canada Geographic Information System* (CGIS), para analizar y visualizar la información del proyecto CLI, desarrollándose así el primer SIG informatizado del mundo [65]. El proyecto CGIS fue totalmente revolucionario, y ha dado lugar a una industria que hoy en día mueve enormes cantidades de dinero en todo el mundo.

Roger Tomlinson inició, planeó y dirigió el proyecto CGIS, y por ello está considerado el "padre" de los SIG actuales.

2.1.1 SISTEMAS DE INFORMACIÓN GEOGRÁFICA

Una de las definiciones más aceptadas de Sistema de Información Geográfica (SIG) es un conjunto integrado de hardware, software y datos geográficos diseñado para capturar, almacenar, actualizar, manipular, analizar y mostrar cualquier tipo de información geográficamente referenciada [2]. En términos más simples, un SIG combina elementos de cartografía, análisis estadístico y bases de datos. Permite a los usuarios crear búsquedas interactivas, analizar la información espacial, editar datos, mapas, y presentar los resultados de todas estas operaciones a través de una interfaz amigable.

Algunos de sus principales usos son: investigación científica, gestión de recursos,evaluación del impacto ambiental, catastro, planificación urbana, arqueología, cartografía, criminología, historia, ventas, marketing, logística o planificación militar. Una de las características más importantes de un SIG es su capacidad de análisis geoespacial. Entre otras muchas tareas, se podrían resolver las siguientes:

- Averiguar cuántos despachos hay a una determinada distancia de un laboratorio o de una clase, o dónde está la salida de incendios más cercana.
- Calcular el camino más corto entre dos lugares.

2.1.2 REPRESENTACIÓN DE INFORMACIÓN ESPACIAL

Una de las técnicas más simples que puede emplear un robot para navegar por un entorno es el paradigma reactivo, que se basa en comportamientos del tipo estímulo-respuesta, como los que se pueden observar en determinados animales. Gracias a la sencillez de implementación, se logra una capacidad de actuación muy rápida, y la navegación se realiza directamente a partir de las percepciones del robot por medio de comportamientos como la evitación de colisiones o el seguimiento de paredes, entre otros. Una carencia básica de los modelos de navegación reactivos es que no necesitan tener definido un modelo del entorno, por lo que no es posible efectuar otras importantes tareas como la planificación de rutas o la localización, necesarias para lograr una capacidad de navegación más avanzada.

Para poder navegar por su entorno y poder estimar su posición o planificar trayectorias, los robots necesitan disponer de una *representación espacial* del mundo que les rodea. Esta representación —o mapa del entorno— es muy variable, y por el momento no existe ningún estándar

que sea utilizado mayoritariamente en la comunidad robótica. Entre otros factores, esto se debe a que los entornos en los que se puede mover un robot son muy diferentes: domésticos, oficinas, urbanos, campo, desierto, minas bajo tierra, o incluso planetas desconocidos. En función del entorno, los tipos de objetos encontrados pueden ser muy distintos, de modo que en la mayoría de los casos la descripción del mapa se trata de adaptar al entorno concreto, en un intento de optimizar su funcionamiento y los recursos del sistema. Además, dependiendo de los sensores hardware disponibles, se debe decidir qué tipo de objetos se van a detectar, y esto también determina el tipo de mapa a construir.

Los mapas pueden ser métricos, topológicos, híbridos, en dos o tres dimensiones, pueden obtenerse *a priori* a partir de fuentes externas, se pueden crear durante la navegación mediante procedimientos de SLAM, o incluso se podrían definir más clasificaciones. Sin embargo, ante las múltiples opciones disponibles, existen algunas propiedades generales que todo mapa debería poseer:

ESCALABILIDAD: es una de las propiedades más deseables en cualquier representación espacial de un entorno. Consiste en la habilidad de poder crecer en tamaño sin perder la calidad en los servicios ofrecidos, o dicho de otro modo, sin mostrar efectos negativos como un excesivo consumo de memoria o gasto computacional. En cuanto a términos de memoria, la escalabilidad está relacionada con la compacidad de la representación, y se vuelve aún más importante en el caso de equipos de robots trabajando conjuntamente y que necesitan enviar diferentes partes de un mapa a través de una red de comunicaciones. Desde el punto de vista de gasto computacional, la escalabilidad en entornos grandes es un elemento crucial cuando se ejecutan determinadas tareas de navegación que se deben ejecutar en tiempo real, como por ejemplo, el seguimiento de trayectorias.

COMPATIBILIDAD CON EL SER HUMANO: una representación del entorno similar a los mapas comprensibles por el ser humano es muy importante, sobre todo en sistemas donde se necesita la interacción hombre-robot, como es el caso de robots guía en museos o ferias, o robot asistentes para tareas domésticas o para los mayores. Un modelo espacial organizado jerárquicamente y que acepte información semántica es mucho más fácil de interpretar por las personas, porque es nuestro modo habitual de comprender la información.

INTEROPERABILIDAD: se puede definir como la capacidad de una representación espacial de ser utilizada e interpretada por diferentes robots con diferentes sistemas sensoriales y diferentes métodos de navegación. Esto está relacionado en cierta manera con la propiedad anterior y se plantea en contraste a los mapas creados específicamente para un robot particular, sus sensores y sus tareas de navegación concretas.

Los Sistemas de Información Geográfica actuales se ajustan bastante bien a las propiedades recién descritas:

- Son *escalables*, dado que almacenan los datos en sistemas de bases de datos espaciales (ver fig. 2.1.7c), que están implementadas indexando los datos espaciales para intentar conseguir la máxima

eficiencia en cuanto a las consultas. Además, los mapas se pueden almacenar en formato vectorial, con lo que la escalabilidad con respecto a la memoria está garantizada.

- Son *compatibles con el ser humano*, puesto que disponen de interfaces que han evolucionado ya a lo largo de los años, y que a pesar de que en sus comienzos eran meramente textuales, hoy en día constituyen una potente herramienta de interacción hombre-máquina.
- Son *interoperables*, porque los datos almacenados en las bases de datos de los SIG siguen la especificación Simple Features Access (SFA) que es tanto un estándar del Open Geospatial Consortium (OGC) como un estándar ISO. Éste estándar define un modelo común de almacenamiento de datos geográficos (puntos, líneas, polígonos, etc.) en formato texto (WKT) o binario (WKB). Esto permite tener un acceso a los datos interoperable y que diferentes robots obtengan información a través de un mismo lenguaje de consulta a la base de datos, conociendo el formato en el que están almacenados los datos.

2.1.3 DATOS ESPACIALES

Los datos espaciales también se conocen habitualmente como *datos geográficamente referenciados* o *datos geoespaciales*. Es un tipo de información que puede ser visualizada, manipulada y analizada en una base de datos gracias a un atributo espacial que indica su ubicación en el mundo. Este atributo espacial normalmente está definido mediante pares de coordenadas que permiten definir la forma y la posición de una determinada característica espacial.

Los datos espaciales hacen referencia a un espacio geográfico, lo que significa que los datos están ubicados en un sistema geográfico de un determinado lugar de la superficie de la Tierra. Gracias a esto, datos obtenidos por distintos medios se pueden interrelacionar e integrar en el espacio. A la hora de almacenar datos espaciales para su posterior análisis y tratamiento, existen dos formatos fundamentales: *vectorial* y *rasterizado*.

2.1.3.1 *Datos rasterizados*

Una de las maneras de obtener datos espaciales *rasterizados* es digitalizando un plano en papel a través de un proceso de escaneado (ver fig. 2.1.3). La imagen obtenida se denomina rasterizada compuesta por una rejilla rectangular de píxeles en 2D que representan la información de una parte del mundo.

El término 'raster' tiene su origen en el vocablo latín 'rastrum' (rastrillo). Se comenzó a usar para designar el trazado línea por línea de las imágenes formadas en monitores CRT. Por asociación se adoptó este término para referirse a las rejillas rectangulares de píxeles.

En el formato rasterizado, la unidad de almacenamiento espacial toma la forma de un *pixel*, que es la mínima unidad de información en una imagen. Esta unidad espacial sirve simultáneamente como un almacén de datos, conteniendo información del espacio que representa, y también como un elemento de referencia geográfica, dado que un píxel dentro de una imagen rasterizada implícitamente denota una ubicación en el mundo real. El tamaño de un píxel en el caso de los datos rasterizados define la resolución, es decir, la cantidad de información del mundo real que representa. Una mayor resolución conlleva una

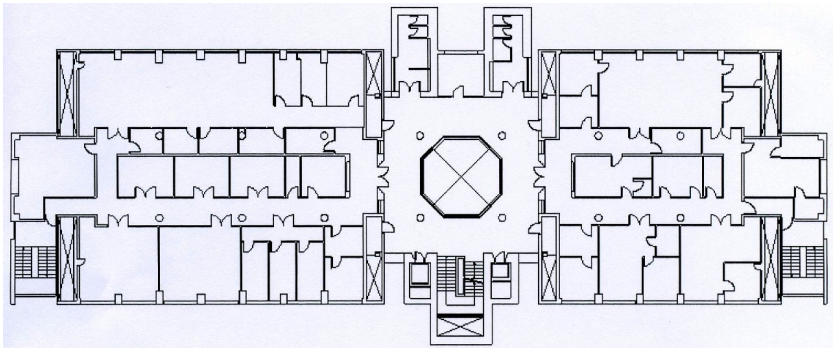


Figura 2.1.3: Imagen rasterizada del plano de la planta baja de la Facultad de Farmacia de la Universidad de Salamanca.

mejor representación del mundo real, pero también implica que la información ocupa mucho más espacio de almacenamiento.

2.1.3.2 *Datos vectoriales*

En una base de datos espacial, la unidad de almacenamiento espacial para los datos en formato vectorial es un objeto geométrico, como se explicará en el apartado 2.1.4, y que identifica una característica del mundo real mediante un punto, una línea, un polígono o una combinación de estos elementos.

Los datos vectoriales se pueden obtener a partir de datos rasterizados, pero esta conversión es un proceso costoso puesto que se deben extraer los objetos geométricos a partir de una imagen rasterizada. Otra manera de crear un plano es diseñarlo directamente en un ordenador en formato vectorial con aplicaciones CAD, en lugar de convertirlos a partir de una imagen rasterizada. De este modo se elimina la etapa de detectar objetos geométricos en el plano, ya que se encuentran definidos nativamente como puntos, líneas o polígonos, en lugar de como píxeles.

Cuanta más abstracción geométrica se incluye en un modelo, más fácil es de interpretar y tomar decisiones usándolo. Por ejemplo, un plano en formato vectorial puede estar formado por un conjunto de líneas sin significado, o por el contrario puede estar formado por polígonos cerrados, que contienen una mayor información semántica, como se muestra en la fig. 2.1.4. En la parte de la izquierda se observa

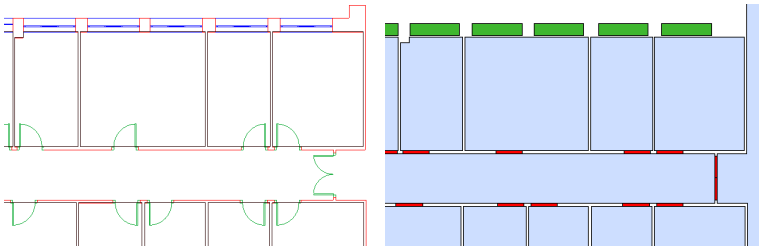


Figura 2.1.4: Dos modelos vectoriales de un mismo entorno, utilizando distinto nivel de abstracción.

una representación de datos vectoriales con elementos codificados por color en diferentes capas: tabiques en rojo, puertas en verde, ventanas en azul, y habitaciones en negro. A pesar del código de colores, puede ser difícil distinguir de qué objeto forman parte las líneas o el espacio entre

líneas, puesto que las líneas no están cerradas formando polígonos. En la parte de la derecha se representa exactamente el mismo entorno, pero cada espacio está modelado mediante un polígono cerrado: las puertas en rojo, las ventanas en verde, y las habitaciones en azul. Al utilizarse polígonos cerrados, una habitación ya no está definida simplemente por sus paredes, sino que cualquier punto dentro del polígono también se considera parte de la habitación. Este tipo de información es sumamente útil para poder conocer, por ejemplo, en qué habitación se encuentra un robot móvil, conociendo simplemente sus coordenadas.

2.1.4 MODELO DE OBJETOS GEOMÉTRICOS

Al hablar de geometría normalmente se hace referencia a una rama de las matemáticas que estudia las propiedades y relaciones de puntos, líneas, ángulos, superficies, sólidos y el espacio en general. Sin embargo, en el contexto del procesamiento de datos espaciales, el concepto de *geometría* tiene un nuevo significado que surgió a raíz de que el *Open Geospatial Consortium* (OGC) publicara la especificación *OpenGIS Simple Feature Specification for SQL* [1]. En este documento, el OGC propone una jerarquía de tipos de datos espaciales denominada *Geometry Object Model* (ver figura 2.1.5), que permite utilizar características espaciales en una base de datos. El término *Geometry* (geometría u objeto geométrico) se utiliza para representar una característica espacial como un *objeto*, utilizando para ello un atributo de tipo "Geometry" en la base de datos. Además, cada geometría está asociada con un sistema de referencia espacial que describe el espacio de coordenadas en que el objeto geométrico está definido.

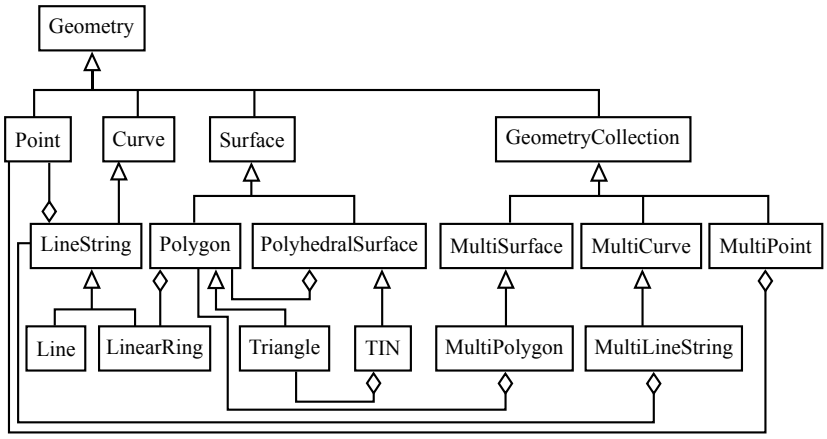


Figura 2.1.5: Jerarquía de clases del *Geometry Object Model*.

La clase Geometry es la clase raíz de la jerarquía de tipos de datos espaciales, es abstracta y no se puede instanciar. Tiene cuatro subclases: Point, Curve, Surface y GeometryCollection, que sí se pueden instanciar y contienen un conjunto de métodos que permiten obtener sus propiedades geométricas y definir sus relaciones espaciales. El resto de tipos geométricos en el modelo permiten construir objetos geométricos más complejos a partir de geometrías más simples. Por ejemplo, las clases MultiPoint y MultiLineString modelan geometrías correspondientes a colecciones de puntos y de líneas, respectivamente.

Puesto que las geometrías básicas están georeferenciadas con respecto a un sistema de referencia particular, la posición y orientación de un

objeto geométrico siempre son conocidas y fijas con respecto a un lugar del espacio concreto. Esto permite establecer relaciones espaciales entre unas geometrías y otras.

Además de su ubicación, una geometría tiene otros atributos que describen sus características, como nombre, dimensión, etc. Los objetos geométricos que comparten los mismos atributos forman una capa *(layer)* y se almacenan en la misma tabla de la base de datos. En la figura 2.1.6 se muestra la relación entre la clase geometría y otros elementos espaciales. Por ejemplo, una capa puede contener todas las puertas de un edificio, mientras que una geometría de esa capa sería una sola puerta. A su vez, cada puerta está definida como una colección de primitivas básicas como son puntos, líneas o polígonos. Finalmente cada primitiva básica consiste en un conjunto de coordenadas métricas o de latitud y longitud.

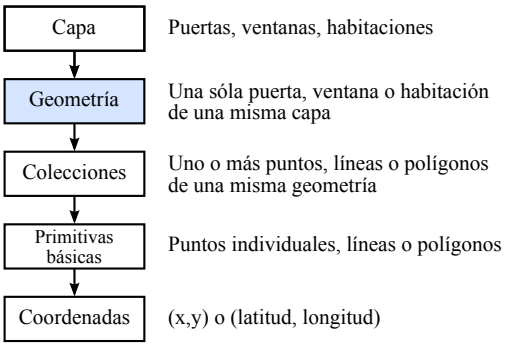


Figura 2.1.6: El concepto de "geometría" y su relacion con otros conceptos espaciales.

La definición de *geometría* del OGC es ampliamente utilizada en la industria de las bases de datos espaciales, aunque la implementación concreta del estándar varía entre los diferentes fabricantes. Gracias a este estándar, definiendo columna de una base de datos relacional convencional como un tipo de datos abstracto, se pueden manejar diferentes objetos geométricos de manera bastante similar a como se manejan los datos alfanuméricos tradicionales. Las bases de datos cuya funcionalidad se extiende para manejar geometrías se denominan bases de datos espaciales, y necesitan una funcionalidad especial para manejar estos tipos de datos más complejos.

2.1.5 BASES DE DATOS ESPACIALES

Según Güting [49], las bases de datos espaciales deben cumplir las siguientes tres características:

1. Un sistema de bases de datos espacial es un sistema de bases de datos.
2. Ofrece Tipos de Datos Espaciales (SDT, *Spatial Data Types)* tanto en el modelo de datos como en el lenguaje de consulta.
3. Soporta SDT en su implementación, proporcionando indización espacial y algoritmos efecientes para los *join* espaciales.

La primera característica parece trivial pero enfatiza que la información geométrica y espacial debe estar conectada con datos tradicionales o

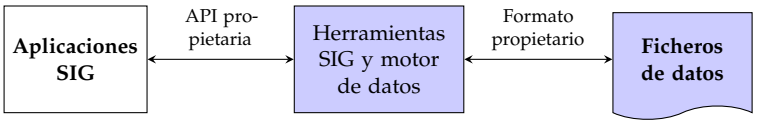
alfanuméricos. Es decir, una base de datos espacial que no tiene las características de una base de datos convencional, carece de todo interés. Así un sistema de bases de datos espacial debe ser una extensión de un sistema de bases de datos convencional. La segunda característica, SDT, se refiere a los puntos, líneas y polígonos descritos en la sección 2.1.4, y que representan objetos geométricos en el espacio así como sus interrelaciones (p.ej. saber si un objeto es adyacente a otro), propiedades (p.ej. el área de un objeto) y operaciones (p.ej. obtener la intersección de dos objetos). Un sistema sin SDT no podrá modelar la información espacial adecuadamente. La última característica indica que un sistema de bases de datos espacial debe ser capaz de acceder a una parte concreta de los datos sin necesidad de acceder uno por uno a todos los datos del conjunto, incluso cuando la base de datos sea muy grande. Al igual que las bases de datos convencionales utilizan índices y la operación de unión *(join)*, las bases de datos espaciales utilizan sus equivalentes (aunque mucho más complejos): índices espaciales y operaciónes de unión *(join)* espaciales.

2.1.5.1 *Evolución y relación con los SIG*

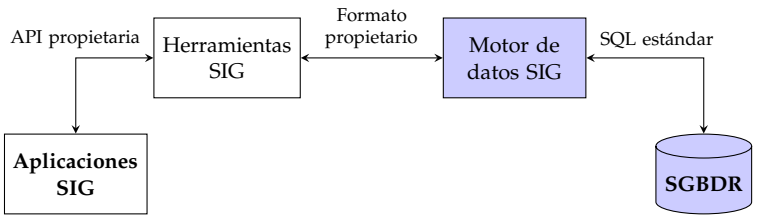
En general, es un hecho aceptado que hoy en día los sistemas de bases de datos espaciales se consideran como la tecnología subyacente de los Sistemas de Información Geográfica (SIG). Por tanto, las bases de datos espaciales no se entienden como un sistema alternativo a los SIG, sino como un sistema que proporciona un subconjunto de la funcionalidad total que ofrece un SIG. A medida que las bases de datos espaciales se fueron desarrollando, la relación entre éstas y los sistemas de información geográfica se fue adaptando hasta llegar a la interrelación y división de tareas actual.

En la fig. 2.1.7 se ilustra la evolución de las técnicas de procesamiento de datos espaciales desde que se comenzaron a desarrollar los primeros SIG comerciales en torno a 1970 [114]. Hasta mediados de la década de 1990 (fig. 2.1.7a), el procesamiento de datos espaciales estaba caracterizado porque los SIG utilizaban una API propietaria y almacenaban los datos en ficheros con formato propietario. Pronto surge la necesidad de manejar volúmenes cada vez mayores de datos espaciales, y esto, junto con avances en la tecnología de bases de datos, conduce a que se comiencen a usar sistemas de gestión de bases de datos para aplicaciones SIG (fig. 2.1.7b). Sin embargo, las bases de datos relacionales se utilizaban como un simple almacén de atributos, y se comunicaban con los SIG por medio de SQL convencional, de modo que todo el procesamiento de datos se realizaba todavía en el SIG. La llegada de los Sistemas de Gestión de Bases de Datos Objeto-Relacionales (SGBDOR) a finales de los 90 hizo posible utilizar la funcionalidad genérica de los SGBDR para procesar datos espaciales para aplicaciones de usuario (fig. 2.1.7c). Además, al usar extensiones de SQL para manejar datos espaciales, un SGBDR puede ser usado no sólo para almacenar atributos sino para almacenar también datos espaciales.

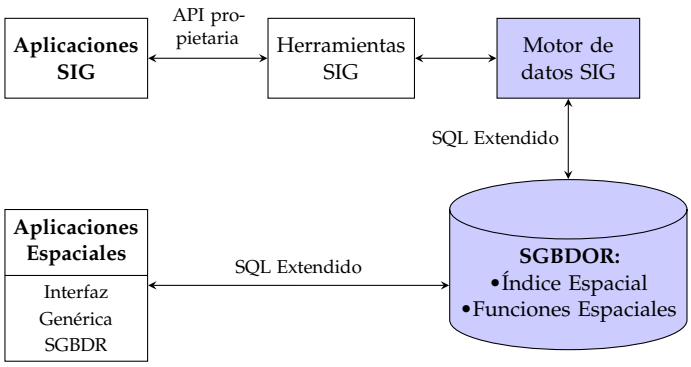
En los sistemas de procesamiento de datos espaciales actuales, la división de tareas entre las bases de datos espaciales y los sistemas de información geográfica es bastante clara (ver tabla 2.1.1). Los sistemas de bases de datos espaciales están implementados de manera que pueden almacenar y manejar grandes volúmenes de información georeferenciada. Estos sistemas indexan los datos espaciales para que



(a) Procesamiento de datos espaciales hasta mediados de los 90.



(b) Procesamiento de datos espaciales a finales de los 90.



(c) Entornos actuales de procesamiento de datos espaciales.

Figura 2.1.7: Evolución del procesamiento de datos espacial.

Sistemas de Información Geográficos	Bases de Datos Espaciales
• Obtención y edición de datos	• Gestión y almacenamiento de datos
• Análisis de datos	• Indexado espacial
• Generación de mapas	• Integridad y seguridad de datos
• Visualización de información	• Consultas de datos espaciales

Tabla 2.1.1: División de tareas entre SIG y bases de datos espaciales.

las consultas sean muy eficientes, y proporcionan los mecanismos y procedimientos necesarios para proteger los datos de destrucción física y pérdida o degradación de integridad lógica. Sin embargo, las bases de datos espaciales generalmente tienen una capacidad limitada para obtener y editar datos, para hacer análisis de datos espaciales, o para generar mapas y otro tipo de información gráfica. Por ello, estas funciones se llevan a cabo por los SIG.

2.1.6 PROCESAMIENTO DE DATOS ESPACIALES

Las consultas de datos espaciales son más complejas que las consultas convencionales porque manejan datos bidimensionales o tridimensionales, y se necesita utilizar índices espaciales y uniones (*join*) espaciales. Una consulta de datos espacial se formula utilizando uno o más operadores que expresan relaciones espaciales. Para poder construir consultas apropiadamente es importante conocer los diferentes opera-

dores. Existen diferentes clasificaciones de los operadores espaciales, y a continuación se muestra la clasificación ofrecida por el OGC [1].

2.1.6.1 *Operadores básicos*

Permiten al usuario acceder a las propiedades generales de una geometría, tales como su ubicación o su forma:

- SRID: devuelve el identificador del sistema de referencia espacial para una geometría.
- Envelope: devuelve el mínimo rectángulo envolvente de una geometría.
- AsText: exporta una geometría al formato *Well-Known Text* (WKT).
- IsEmpty: comprueba si una geometría está formada por el conjunto vacío.
- IsSimple: comprueba si una geometría no tiene puntos geométricos anómalos, tales como autotangencias o autointersecciones.

2.1.6.2 *Operadores topológicos*

Son un conjunto de métodos que ejecutan diferentes pruebas espaciales entre dos objetos geométricos:

- *Equals*: comprueba si dos geometrías son exactamente la misma.
- *Disjoint*: comprueba si dos geometrías no se tocan.
- *Intersects*: comprueba si dos geometrías tienen 1 o más puntos en común. Es lo mismo que "Not Disjoint".
- *Touches*: comprueba si dos geometrías tienen por lo menos un punto en común, pero no en el interior de las geometrías.
- *Crosses*: comprueba si la intersección de dos geometrías no es el conjunto vacío y da como resultado una geometría de dimensión menor.
- *Within*: Comprueba si una geometría está dentro de otra diferente.
- *Contains*: Comprueba si una geometría tiene dentro otra distinta.
- *Overlaps*: Comprueba si la intersección de dos geometrías da como resultado una geometría distinta pero de la misma dimensión.

En la figura 2.1.8 se muestran algunas posibles interpretaciones de estos operadores.

2.1.6.3 *Operadores de análisis espacial*

Permiten al usuario construir consultas espaciales más avanzadas que las ofrecidas por los operadores topológicos:

- *Distance*: devuelve la distancia más corta entre cualquier par de puntos de dos geometrías.

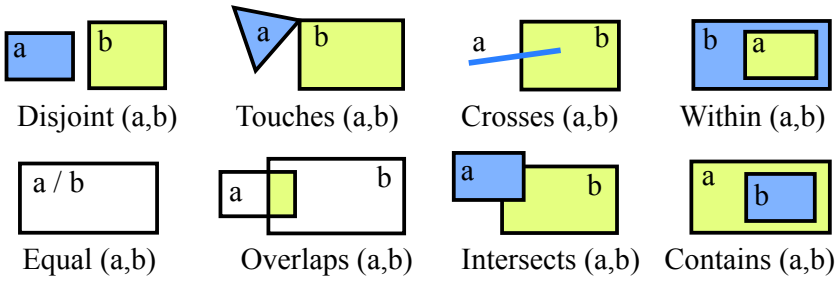


Figura 2.1.8: Algunos ejemplos concretos de diferentes relaciones topológicas entre dos geometrías distintas.

- *Buffer*: devuelve una geometría que representa todos los puntos a una distancia menor o igual que la distancia especificada.
- *ConvexHull*: devuelve la envolvente convexa de una geometría.
- *Intersection*: devuelve una geometría que representa la intersección del conjunto de puntos de dos geometrías.
- *Union*: devuelve una geometría que representa la unión del conjunto de puntos de dos geometrías.
- *Difference*: devuelve una geometría que representa la parte de una geometría que no forma intersección con otra.
- *SymDifference*: devuelve una geometría que representa las porciones de dos geometrías que no forman intersección entre ellas.

CAPÍTULO 2.2

RESUMEN

Esta parte comienza con el estudio de la necesidad de almacenar y representar la información espacial, y de cómo las técnicas utilizadas para afrontar esta necesidad han ido evolucionando en el tiempo hasta dar lugar a los Sistemas de Información Geográfica actuales. Los SIG están diseñados con el objetivo de almacenar, manipular, y visualizar cualquier tipo de información geográficamente referenciada, y sus usos son tan variados que abarcan desde la criminología o la planificación militar hasta la arqueología, pasando por la cartografía, la logística o la investigación científica, entre otras muchas posibilidades.

Para un robot autónomo es muy importante disponer de una representación espacial de su entorno, pero hasta el momento no existe ningún estándar de modelado espacial que sea utilizado mayoritariamente para esta tarea. A pesar de esto, se puede suponer que cualquier representación del espacio debería cumplir con tres requisitos generales: escalabilidad, compatibilidad con el ser humano, e interoperabilidad. Precisamente estas tres propiedades son características de los SIG, motivo por el que resultan muy interesantes para su uso en robótica móvil.

Una característica destacada de los SIG es que en la última década su evolución les ha llevado a integrarse perfectamente con las bases de datos espaciales. Estas bases de datos son similares a las convencionales, con la diferencia de que incorporan una extensión que les permite almacenar y manejar tipos de datos espaciales, así como optimizar las consultas por medio de índices espaciales, evitando tener que acceder a todo el conjunto de datos.

Para definir los tipos de datos espaciales, las bases de datos espaciales utilizan un estándar del *Open Geospatial Consortium* en el que se especifica un *Modelo de Objetos Geométricos*. Gracias a este modelo se puede representar cualquier característica del entorno por medio de una jerarquía de objetos. Además las características se pueden separar por capas, facilitando así el manejo de la información simbólica. El acceso a los datos se logra por medio de la construcción de consultas espaciales, gracias a un conjunto de operadores topológicos y de análisis espacial (como unión, diferencia, etc.) definidos en el estándar del OGC anteriormente citado.

Parte 3

ESTIMACIÓN DE SISTEMAS ESTOCÁSTICOS

CAPÍTULO 3.1

EL FILTRO DE KALMAN (KF)

El filtro de Kalman se puede definir de forma muy concisa como un algoritmo de procesamiento de datos recursivo que estima el estado de un sistema dinámico lineal perturbado por ruido basándose en observaciones periódicas del estado perturbadas por ruido [69, 98]. Es habitual encontrar la denominación *sistemas estocásticos lineales* cuando se hace referencia a los sistemas dinámicos lineales perturbados por ruido [11, 46]. Un aspecto a destacar del filtro de Kalman es que se trata de un filtro de naturaleza probabilística, dado que es capaz de manejar incertidumbre en las estimaciones, y se puede considerar como una instancia específica del *filtro de Bayes recursivo* cuando las variables son lineales y responden a una distribución normal [106]. El hecho de que sea recursivo implica que no requiere almacenar permanentemente todas las observaciones y estimaciones previas para ser procesadas de nuevo cada vez que se realiza una nueva observación, siendo esto de vital importancia a la hora de implementar el filtro en la práctica.

El estado de un sistema consiste en un conjunto mínimo de variables que describen las propiedades del sistema que se desean observar. Por ejemplo, en el caso de un robot es habitual definir el estado como su posición y orientación con respecto a un sistema de referencia global. En muchas ocasiones el valor real del estado no se puede conocer directamente, lo que hace que el problema de estimación sea bastante complejo. Sin embargo, utilizando sensores se pueden obtener datos que estén relacionados funcionalmente con el estado del sistema.

Para efectuar la estimación del estado el filtro de Kalman procesa periódicamente todas las observaciones disponibles sea cual sea la precisión con la que se hayan efectuado, utilizando además:

- Un modelo estadístico explícito de cómo el estado evoluciona en el tiempo (*modelo del sistema*) y un modelo estadístico explícito de cómo las observaciones están relacionadas con el estado (*modelo de observación*).
- Una descripción estadística de las perturbaciones por ruido que afectan al sistema, los errores en las observaciones y la incertidumbre en los modelos.
- Cualquier información disponible acerca de las condiciones iniciales de las variables de estado.

El filtro de Kalman tiene una serie de características que lo hacen especialmente apropiado para resolver problemas de fusión de datos y estimación multisensorial. La descripción explícita del modelo del sistema y del modelo de observación permiten incorporar en el algoritmo básico una amplia variedad de modelos sensoriales diferentes. Además, el uso consistente de medidas estadísticas de incertidumbre

hace posible la evaluación cuantitativa del papel que desempeña cada sensor en el rendimiento del sistema, y la naturaleza lineal recursiva del algoritmo garantiza que su aplicación sea simple y eficiente. Por estos motivos el filtro de Kalman ha encontrado una aplicación generalizada en problemas de fusión sensorial en ámbitos muy diversos.

3.1.1 SUPOSICIONES

El filtro de Kalman supone que el sistema a modelar es estocástico y lineal, que el ruido afecta al sistema cumple ciertas características, así como que se dispone de un modelo del sistema y un modelo de observación.

3.1.1.1 *Sistema estocástico lineal*

Ningún modelo matemático de un sistema real es perfecto, sino que simplemente trata de describir los rasgos dominantes del sistema. Además, los sistemas dinámicos no están afectados solamente por las entradas de control, sino también por perturbaciones que no se pueden controlar ni modelar determinísticamente. Por otra parte, los datos proporcionados por los sensores de un sistema no son perfectos, sino que introducen pequeñas distorsiones. En definitiva, suponer que los modelos matemáticos que describen el sistema son *deterministas* es un enfoque ingenuo e inadecuado [69]. Debido a esto, el filtro de Kalman se basa en modelos lineales que tienen en cuenta las perturbaciones por ruido que afectan a los sistemas del mundo real. Sin embargo muchos sistemas contienen no linealidades, por lo que necesitan linealizarse mediante el Filtro Extendido de Kalman, como se explicará en el apartado 3.2.

3.1.1.2 *Modelo del Sistema*

El modelo del sistema describe cómo el estado *real* del sistema evoluciona en el tiempo, y el filtro de Kalman lo utiliza para poder hacer estimaciones acerca del estado. Utilizando la representación del espacio de estados para un sistema discreto, la evolución del estado $\mathbf{x}$ se describe como:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_k \mathbf{x}_k + \mathbf{B}_k \mathbf{u}_k + \mathbf{v}_k \\ \mathbf{v}_k &\sim \mathcal{N}(0, \mathbf{Q}_k) \end{aligned} \tag{3.1.1}$$

donde $\mathbf{x}_k$ es la variable de estado n-dimensional en el instante de tiempo k, $\mathbf{F}_k$ es la matriz de transición de estados, $\mathbf{u}_k$ la entrada de control al sistema y $\mathbf{v}_k$ una perturbación por ruido.

La notación $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ se emplea para indicar que el ruido que afecta al sistema se supone que es *ruido blanco gaussiano*. Esto significa que la incertidumbre se puede modelar mediante una distribución normal multivariante con media $E[\mathbf{v}(k)] = 0$ y covarianza $\mathbf{Q}_k$, cumpliéndose que $E[\mathbf{v}_k(i)\mathbf{v}_k^T(j)] = \delta_{ij}\mathbf{Q}_k(i)$.

3.1.1.3 *Modelo de observación*

El modelo de observación determina la relación que existe entre las observaciones y el estado del sistema, y es utilizado por el filtro de

Kalman para poder corregir la predicción del estado cada vez que está disponible una nueva observación.

El papel fundamental del modelo de observación es permitir *predecir* el valor de una observación suponiendo que el sistema está en un determinado estado. Gracias a esto se puede comparar *la predicción de la observación* con la observación realizada, y se calcula el error en la observación, que es fundamental para corregir la estimación del estado mediante las observaciones realizadas.

El filtro de Kalman supone que las observaciones se pueden modelar mediante una ecuación que relacione *linealmente* el estado del sistema con las observaciones, de modo que para un sistema discreto, el modelo de observación se puede representar como:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{w}_k \\ \mathbf{w}_k &\sim \mathcal{N}(0, \mathbf{R}_k) \end{aligned} \tag{3.1.2}$$

donde $\mathbf{z}_k$ es un vector de observaciones m-dimensional, $\mathbf{H}_k$ es la matriz de observación, $\mathbf{x}_k$ es el vector de estado, y $\mathbf{w}_k$ es una perturbación por ruido blanco gaussiano, con media $E[\mathbf{w}(k)] = 0$ y covarianza $\mathbf{R}_k$, cumpliéndose que $E[\mathbf{w}_k(i)\mathbf{w}_k^T(j)] = \delta_{ij}\mathbf{R}_k(i)$.

3.1.1.4 *Características del ruido*

El filtro de Kalman supone que tanto el ruido en el modelo del sistema, $\mathbf{v}(k)$, como el ruido en el modelo de observación, $\mathbf{w}(k)$, son variables aleatorias multivariantes e independientes entre sí que modelan ruido blanco gaussiano:

- La suposición de que las fuentes de ruido son *independientes* significa que el ruido que afecta a una determinada variable en un instante de tiempo no afecta a las demás. Por ejemplo, el ruido existente en las intensidades de los píxeles de una imagen tomadas por una cámara de visión no se ve influenciado por el ruido en las medidas de un escáner láser o por el ruido en el sistema de odometría.

- El *ruido blanco* se caracteriza por tener una densidad espectral de potencia constante, lo que significa que en el caso de muestrear una variable aleatoria en dos instantes de tiempo diferentes, sus valores no estarían correlacionados estadísticamente. Esto quiere decir que conociendo la cantidad de ruido en un instante determinado no es posible predecir la cantidad de ruido en el futuro. Además, el ruido blanco tiene *media cero*, lo que implica que los errores en el modelo del sistema y en el modelo de observación son aleatorios, y por lo tanto son errores no sistemáticos y no sesgados. Un ejemplo de error sistemático sería un diámetro de rueda mal calibrado, que afectaría constantemente de manera positiva o negativa, de modo que la media del error estaría sesgada y no sería cero.

- Por otra parte, suponer que la distribución del ruido sea *gaussiana* es la opción más sensata a priori y puede justificarse por el teorema central del límite [88], aduciendo que los ruidos modelados son el resultado de la suma de un gran número de fuentes de error aleatorias. Otra buena razón es que al calibrar los sensores es

normalmente sencillo realizar experimentos para averiguar su media y varianza del error, y precisamente la función de densidad de probabilidad de una distribución gaussiana está completamente descrita por su media y su varianza, por lo que no se necesita calcular estadísticos de orden mayor y se captura toda la información disponible acerca del ruido.

Al incorporar el ruido blanco gaussiano $\mathbf{v}(k)$ a la descripción del modelo de sistema (3.1.1) se logra que el estado $\mathbf{x}_k$ pase automáticamente a responder a una distribución normal. Esto se debe a que una de las propiedades de las distribuciones normales es que si $\mathbf{x}_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$ y $\mathbf{x}_2 \sim \mathcal{N}(\mu_2, \Sigma_2)$ son dos variables aleatorias independientes con una distribución normal multivariante, entonces la suma de ambas seguirá teniendo una distribución normal multivariante:

$$\mathbf{x}_1 + \mathbf{x}_2 \sim \mathcal{N}(\mu_1 + \mu_2,\ \Sigma_1 + \Sigma_2) \tag{3.1.3}$$

Así, teniendo en cuenta que $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ es la parte de la ecuación que representa el ruido, se puede considerar que el resto de la expresión carece de ruido, o lo que es lo mismo, que su covarianza es 0. Por lo tanto se puede considerar que $(\mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k) \sim \mathcal{N}(\mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, 0)$, y utilizando la propiedad (3.1.3), resulta que:

$$\mathbf{x}_{k+1} \sim \mathcal{N}(\mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k, \mathbf{Q}_k) \tag{3.1.4}$$

De manera similar se puede llegar a la conclusión de que el modelo de observación (3.1.2) responde a la distribución normal:

$$\mathbf{z}_k \sim \mathcal{N}(\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k) \tag{3.1.5}$$

3.1.2 FORMULACIÓN DEL PROBLEMA DE ESTIMACIÓN

Suponiendo que se dispone de una descripción del sistema en la forma de la ecuación (3.1.1) y de observaciones discretas y relacionadas con el estado mediante la ecuación (3.1.2), el filtro de Kalman afronta el problema de efectuar una estimación $(\hat{\mathbf{x}})$ acerca del valor real del estado $(\mathbf{x})$. El error en la estimación del estado $(\tilde{\mathbf{x}})$ se define como la diferencia entre el valor real y el valor estimado:

$$\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}} \tag{3.1.6}$$

Una característica fundamental del filtro de Kalman es que no solamente mantiene una estimación del vector de estado $(\hat{\mathbf{x}})$, sino que también mantiene una medida estadística de la incertidumbre asociada a dicha estimación [29]. Esta medida de incertidumbre se calcula como la matriz de covarianza de $\tilde{\mathbf{x}}$, y se representa por $\mathbf{P}$. Por lo tanto, el filtro de Kalman estima el valor real del estado como $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P})$, es decir, como una función de densidad de probabilidad normal multivariante con media $\hat{\mathbf{x}}$ y covarianza $\mathbf{P}$, donde:

$$\hat{\mathbf{x}} = E[\mathbf{x}] = \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \vdots \\ \hat{\mathbf{x}}_n \end{bmatrix} \tag{3.1.7}$$

$$\mathbf{P} = E\left[\tilde{\mathbf{x}} \cdot \tilde{\mathbf{x}}^T\right] = E\left[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T\right] = \begin{pmatrix} \mathbf{P}_{11} & \cdots & \mathbf{P}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{n1} & \cdots & \mathbf{P}_{nn} \end{pmatrix} \tag{3.1.8}$$

En el contexto de la localización de un robot móvil, el hecho de que el filtro de Kalman estime el estado como una distribución de probabilidad gaussiana significa que la ubicación del robot no será una única posición, sino que vendrá expresada como una distribución de posibles ubicaciones del robot con un cierto grado de probabilidad asociada.

3.1.3 ALGORITMO

El filtro de Kalman calcula la estimación del estado ($\hat{\mathbf{x}}$) y su matriz de covarianza ($\mathbf{P}$) basándose en un ciclo de predicción-corrección: en primer lugar predice una estimación del estado utilizando el modelo dinámico del sistema (3.1.1) y a continuación, cuando hay observaciones disponibles, corrige la predicción del estado utilizando el modelo de observación (3.1.2).

La notación $\hat{\mathbf{x}}_{k_1|k_2}$ se utiliza para representar la estimación del estado en el instante de tiempo k_1 teniendo en cuenta las observaciones disponibles hasta el instante de tiempo k_2 incluido. A continuación se muestran las ecuaciones para cada una de las fases sin su demostración matemática correspondiente. Si el lector está interesado en la obtención de las ecuaciones se recomienda que consulte obras de referencia como [11, 44, 69].

PREDICCIÓN: En cada instante de tiempo el estado del sistema puede variar y por ello en cada incremento de tiempo el filtro de Kalman calcula una nueva estimación del estado a partir de la última estimación disponible. Dada la estimación del estado del sistema y su matriz de covarianza $(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$ en el instante de tiempo k, la predicción $(\hat{\mathbf{x}}_{k+1|k}, \mathbf{P}_{k+1|k})$ en el instante de tiempo $k+1$ se calculan como:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{F}_k \cdot \hat{\mathbf{x}}_{k|k} + \mathbf{B}_k \cdot \mathbf{u}_k \tag{3.1.9}$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_k \cdot \mathbf{P}_{k|k} \cdot \mathbf{F}_k^{\top} + \mathbf{Q}_k \tag{3.1.10}$$

CORRECCIÓN: A diferencia de la fase de predicción, la fase de corrección solamente se efectúa cuando están disponibles nuevas observaciones, que proporcionan información directa acerca del estado actual del sistema. Las ecuaciones de corrección actualizan la estimación del estado más reciente mediante la incorporación de la información ganada a partir de las observaciones.

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1} \cdot \tilde{\mathbf{z}}_{k+1} \tag{3.1.11}$$

$$\mathbf{P}_{k+1|k+1} = \left[\mathbf{I} - \mathbf{K}_{k+1} \cdot \mathbf{H}_{k+1}\right] \cdot \mathbf{P}_{k+1|k} \tag{3.1.12}$$

donde (3.1.11) es la *corrección de la estimación del estado*, (3.1.12) es la *corrección de la covarianza del estado*, y

$$\tilde{\mathbf{z}}_{k+1} = \mathbf{z}_{k+1} - \mathbf{H}_{k+1} \cdot \hat{\mathbf{x}}_{k+1|k} \tag{3.1.13}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1} \cdot \mathbf{P}_{k+1|k} \cdot \mathbf{H}_{k+1}^{\top} + \mathbf{R}_{k+1} \tag{3.1.14}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k} \cdot \mathbf{H}_{k+1}^{\top} \mathbf{S}_{k+1}^{-1} \tag{3.1.15}$$

(3.1.13) es la *innovación*, (3.1.14) es la *covarianza de la innovación* y (3.1.15) es la *ganancia óptima de Kalman*, que está elegida de modo que garantiza que la estimación del estado minimiza el error cuadrático medio.

Un aspecto a destacar en la ecuación de innovación (3.1.13) es que el filtro de Kalman efectúa una *predicción de la observación* definida

mediante la expresión: $\mathbf{H}_{k+1} \cdot \hat{\mathbf{x}}_{k+1|k} = \hat{\mathbf{z}}_{k+1}$. Esta predicción indica cuál es el valor que el filtro predice que debería tener la observación $\mathbf{z}_{k+1}$, basándose en la predicción del estado $\hat{\mathbf{x}}_{k+1|k}$. Reescribiendo la ecuación de innovación como $\tilde{\mathbf{z}}_{k+1} = \mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1}$ resulta más evidente que la innovación es una medida del error que el filtro comete al predecir la observación.

Cuando la innovación es próxima a cero entonces la predicción de la observación es muy similar a la observación real, lo que implicaría que la predicción del estado (desde la que se hizo la predicción de la observación) sería ya bastante precisa y no necesitaría una gran corrección. Por el contrario, cuando la innovación es grande, la predicción de la observación difiere bastante de la observación real, y esto quiere decir que la predicción del estado necesitaría una fuerte corrección.

CAPÍTULO 3.2

EL FILTRO EXTENDIDO DE KALMAN (EKF)

Algunas de las aplicaciones de mayor relevancia a la hora de utilizar el filtro de Kalman han sido en situaciones donde la dinámica del sistema o el modelo de observación eran no lineales. Quizás dos de los ejemplos más destacables de un filtro de Kalman no lineal sean el control de la cápsula espacial del programa *Apollo* de la NASA [70], cuyo objetivo era llevar una misión tripulada a la Luna y traerla de regreso a la Tierra sin percance alguno, o su implementación para el cálculo de las trayectorias de las sondas *Voyager* [22], que aún hoy siguen funcionando después de más de 30 años desde su lanzamiento.

Cuando los sistemas con los que se trabaja son robots móviles, suponer que el modelo de sistema y el modelo de observación son lineales con perturbaciones por ruido blanco gaussiano son unos requisitos que difícilmente se pueden cumplir. Basta pensar en un robot con velocidad constante de avance y de rotación para darse cuenta que la trayectoria que describe es circular y no se puede describir con un modelo de sistema lineal. Esto hace que el filtro de Kalman lineal resulte inaplicable a la mayoría de problemas de localización de robots, a excepción de los más triviales [106].

A la hora de aplicar el filtro de Kalman a problemas no lineales existen principalmente dos técnicas. La primera consiste en utilizar una serie de Taylor de primer orden en torno a una trayectoria de referencia *estática*, dando como resultado lo que se conoce como *Filtro de Kalman Linealizado* [116]. Es una técnica útil cuando existe una gran cantidad de información a priori acerca del problema, como por ejemplo en el seguimiento de satélites, ya que se conoce de antemano la órbita aproximada del satélite. El principal problema de este enfoque es que se utiliza la misma trayectoria de referencia mientras dura el proceso de estimación, y los errores crecerían sin límite en el caso de que haya una gran diferencia entre el estado del sistema y la trayectoria de referencia. Para evitar este problema, la solución consiste en linealizar mediante una serie de Taylor en torno a una trayectoria de referencia *dinámica* que se está actualizando constantemente con estimaciones del estado que son producto de las observaciones. Es decir, cada vez que se calcula una nueva estimación del estado, se recalcula una nueva trayectoria de referencia mejor y más precisa y se incorpora al proceso de estimación. Cuando se utiliza una serie de Taylor de primer orden, el filtro resultante se conoce como *Filtro Extendido de Kalman* (EKF) [69], mientras que si se utiliza una serie de Taylor de segundo orden se obtiene el *EKF de segundo orden* [11]. En este trabajo se utilizará exclusivamente el EKF estándar o de primer orden, basados en la suposición de un modelo de sistema no lineal y un modelo de observación no lineal, cuyas ecuaciones se describen a continuación.

Una 'trayectoria de referencia' se define como una solución específica de un sistema estocástico.

El término 'EKF' corresponde a las siglas en inglés: 'Extended Kalman Filter'.

3.2.1 MODELO DEL SISTEMA

En el EKF, el sistema tiene que estar representado por una ecuación *no lineal* de la forma:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_k \\ \mathbf{v}_k &\sim N(0, \mathbf{Q}_k) \end{aligned} \tag{3.2.1}$$

donde $\mathbf{f}(\cdot)$ es una *función de transición de estados no lineal* y $\mathbf{v}_k$ representa una perturbación por ruido blanco gaussiano, con media $E[\mathbf{v}(k)] = 0$ y covarianza $\mathbf{Q}_k$, cumpliéndose que $E[\mathbf{v}_k(i)\mathbf{v}_k^T(j)] = \delta_{ij}\mathbf{Q}_k(i)$.

3.2.2 MODELO DE OBSERVACIÓN

Igual que en el modelo de sistema, el modelo que relaciona las observaciones con el estado también pasa a estar expresado mediante una ecuación *no lineal*:

$$\begin{aligned} \mathbf{z}_k &= \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k \\ \mathbf{w}_k &\sim N(0, \mathbf{R}_k) \end{aligned} \tag{3.2.2}$$

donde $\mathbf{h}(\cdot)$ es una *función de observación no lineal* y $\mathbf{w}_k$ es una perturbación por ruido blanco gaussiano, con media $E[\mathbf{w}(k)] = 0$ y covarianza $\mathbf{R}_k$, cumpliéndose que $E[\mathbf{w}_k(i)\mathbf{w}_k^T(j)] = \delta_{ij}\mathbf{R}_k(i)$.

3.2.3 ALGORITMO

De manera similar al filtro de Kalman lineal, el algoritmo del EKF se basa en un ciclo de predicción-corrección, con la diferencia de que se linealizan las funciones no lineales mediante una serie de Taylor de primer orden. A continuación se muestran las ecuaciones sin su desarrollo matemático. El lector interesado puede encontrar diversas demostraciones en [11, 69, 106] entre otras.

PREDICCIÓN: Dada la estimación del estado del sistema y su matriz de covarianza $(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$ en el instante de tiempo k, la predicción $(\hat{\mathbf{x}}_{k+1|k}, \mathbf{P}_{k+1|k})$ en el instante de tiempo $k+1$ se calcula como:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) \tag{3.2.3}$$

$$\mathbf{P}_{k+1|k} = \nabla_x \mathbf{f} \cdot \mathbf{P}_{k|k} \cdot \nabla_x \mathbf{f}^T + \mathbf{Q}_k \tag{3.2.4}$$

donde el Jacobiano de la función de transición de estados $(\nabla_x \mathbf{f})$ evaluada en el estado $\mathbf{x} = \hat{\mathbf{x}}_{k|k}$ se define como:

$$\nabla_x \mathbf{f} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k}} = \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{f}_n}{\partial \mathbf{x}_n} \end{bmatrix}_{\mathbf{x}=\hat{\mathbf{x}}_{k|k}} \tag{3.2.5}$$

CORRECCIÓN: En el instante de tiempo $k+1$ se realiza la observación $\mathbf{z}_{k+1}$, y utilizando la predicción del estado calculada en la fase anterior, se corrige la estimación del estado de acuerdo a las ecuaciones:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1} \cdot \tilde{\mathbf{z}}_{k+1} \tag{3.2.6}$$

$$\mathbf{P}_{k+1|k+1} = \left[I - \mathbf{K}_{k+1} \cdot \nabla_x \mathbf{h}\right] \mathbf{P}_{k+1|k} \tag{3.2.7}$$

donde (3.2.6) es la *corrección de la estimación del estado*, y (3.2.7) es la *corrección de la covarianza del estado*, y los términos:

$$\tilde{\mathbf{z}}_{k+1} = \mathbf{z}_{k+1} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) \tag{3.2.8}$$

$$\mathbf{S}_{k+1} = \nabla_x \mathbf{h} \cdot \mathbf{P}_{k+1|k} \cdot \nabla_x \mathbf{h}^T + \mathbf{R}_{k+1} \tag{3.2.9}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k} \cdot \nabla_x \mathbf{h}^T \cdot \mathbf{S}_{k+1}^{-1} \tag{3.2.10}$$

representan la *innovación* (3.2.8), la *covarianza de la innovación* (3.2.9), y la *ganancia subóptima (o quasi-óptima) de Kalman* (3.2.10), y la función de observación se linealiza mediante el Jacobiano ($\nabla \mathbf{h}_x$), evaluado utilizando la última predicción del estado $\mathbf{x} = \hat{\mathbf{x}}_{k+1|k}$:

$$\nabla_x \mathbf{h} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k+1|k}} = \left[\begin{array}{ccc} \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{h}_1}{\partial \mathbf{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{h}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{h}_n}{\partial \mathbf{x}_n} \end{array}\right]_{\mathbf{x}=\hat{\mathbf{x}}_{k+1|k}} \tag{3.2.11}$$

Comparando las ecuaciones de predicción-corrección del filtro de Kalman lineal (3.1.9-3.1.15) con las del EKF (3.2.3-3.2.11) se puede apreciar que ambas son muy similares, con las siguientes diferencias:

1. Se han sustituido funciones lineales ($\mathbf{F}_k, \mathbf{B}_k, \mathbf{H}_k$) por funciones no lineales ($\mathbf{f}(\cdot), \mathbf{h}(\cdot)$) en el modelo del sistema y el modelo de observación.

2. Se han sustituido las matrices $\mathbf{F}_k$ y $\mathbf{H}_k$ por los Jacobianos $\nabla \mathbf{f}$ y $\nabla \mathbf{h}$ en el cálculo de la covarianza y la ganancia del filtro.

Normalmente los Jacobianos $\nabla \mathbf{f}$ y $\nabla \mathbf{h}$ no son constantes sino que varían en función del estado y del instante de tiempo, de modo que la matriz de covarianza y la matriz de ganancia se deben recalcular continuamente, incrementándose así los requisitos computacionales necesarios.

3.2.4 COMPENSACIÓN HEURÍSTICA DE LOS ERRORES

Dado que el EKF se obtiene mediante una linealización basada en una serie de Taylor de primer orden, se están introduciendo errores en los cálculos, debido principalmente a que [11]:

- Los términos de segundo orden o mayores en la serie de Taylor no se están teniendo en cuenta.
- Las evaluaciones de los Jacobianos se están haciendo utilizando valores estimados del estado en lugar de utilizar los valores reales, dado que éstos no están disponibles.

Para compensar estos errores, que pueden llevar a la *divergencia* del filtro, se pueden utilizar los siguientes métodos heurísticos:

- Añadir *ruido artificial* o *pseudo-ruido* al modelo del sistema para compensar los errores en la predicción del estado. Esto se puede lograr utilizando una matriz de covarianza del sistema modificada $\mathbf{Q}_k^m$ en la ecuación de predicción de la covarianza (3.2.4):

  $$\mathbf{Q}_k^m = \mathbf{Q}_k^p + \mathbf{Q}_k \geqslant \mathbf{Q}_k \tag{3.2.12}$$

  donde $\mathbf{Q}_k^p$ es una matriz de covarianza con ruido artificial.

- Multiplicar la covarianza del estado por un escalar $\phi > 1$ en cada tiempo de muestreo:

$$\mathbf{P}^{\phi}_{k+1|k} = \phi \cdot \mathbf{P}_{k+1|k} \tag{3.2.13}$$

y utilizar esta matriz modificada en la ecuación de predicción de la covarianza (3.2.4). La multiplicación de la matriz de covarianza por un escalar mayor que la unidad es equivalente a un filtro que pierde la memoria de lo sucedido, ya que es como si las últimas actualizaciones de la covarianza se hubiesen perdido, ya que la incertidumbre se hace mayor.

Mediante cualquiera de estas dos técnicas se consigue que la matriz de covarianza "cubra" los errores cometidos al linealizar un sistema no lineal. Al incrementar el error en la matriz de covarianza del estado ($\mathbf{Q}_k$) se consigue que la ganancia del filtro sea mayor, dándole mayor peso a las observaciones más recientes. Si por el contrario se incrementa el error en la matriz de covarianza de las observaciones ($\mathbf{R}_k$), se obtendría el efecto opuesto: bajar la ganancia del filtro y ponderar en menor medida las nuevas observaciones.

CAPÍTULO 3.3

MODELO DEL SISTEMA

Para resolver el problema de localización (ver capítulo 5.5) de nuestro robot móvil, en este trabajo se utiliza el filtro extendido de Kalman (EKF) como herramienta matemática para fusionar la información odométrica y las observaciones realizadas por el sistema sensorial del robot. Como se explicó en el capítulo 3.1, el EKF se basa en el uso de un modelo de sistema y un modelo de observación que se han de definir meticulosamente en función del problema específico que se pretenda resolver.

En este capítulo se desarrolla el *modelo del sistema*, que sirve para calcular la ubicación del robot en función de las medidas de desplazamiento relativas proporcionadas por sus *encoders*. A continuación, en el capítulo 3.4 se define el *modelo de observación*, que permite relacionar las observaciones realizadas por el robot con su estado.

3.3.1 MODELO DEL SISTEMA

Al aplicar el EKF al problema de localización de nuestro robot móvil, en primer lugar se necesita especificar un modelo de sistema que responda a la ecuación:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{v}_k \tag{3.3.1}$$

A continuación se especifican las variables de estado ($\mathbf{x}$), así como la función de transición de estados ($\mathbf{f}$) y el modelo del ruido ($\mathbf{v}$) que afecta al robot.

3.3.2 VECTOR DE ESTADO

El estado se ha definido empleando variables que permitan localizar al robot en el espacio de manera única. Considerando un sistema de referencia *global* cartesiano bidimensional, el estado se ha determinado utilizando:

1. Las coordenadas (x, y) de un punto del robot que lo identifique en el espacio cartesiano.
2. La orientación del robot (θ), para indicar la dirección de avance del robot en el sistema de referencia.

Añadir una tercera dimensión resulta innecesario, dado que en entornos estructurados generalmente no existen variaciones significativas de altitud, por lo que un espacio cartesiano bidimensional es suficiente.

Así, utilizando un sistema de tiempos discreto, el estado del robot ($\mathbf{x}$) en el instante de tiempo k está dado por:

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{3.3.2}$$

En el modelo del sistema, los cambios en el estado del robot, es decir, cambios de posición u orientación, se miden mediante *encoders* incrementales, que proporcionan medidas de desplazamiento relativas con respecto al sistema de referencia local del robot. Los incrementos de posición interpretados de manera aislada no resultan de ninguna utilidad. Sin embargo, si se integran en el tiempo, permiten conocer la trayectoria del robot en el sistema de referencia global, siempre y cuando se conozca la posición de partida. Para llevar a cabo esta tarea, y convertir los giros o traslaciones medidos en el sistema de referencia local del robot $\{X_R, Y_R\}$ al sistema de referencia global $\{X_G, Y_G\}$, se debe definir claramente la relación entre ambos sistemas, como se muestra en la figura 3.3.1.

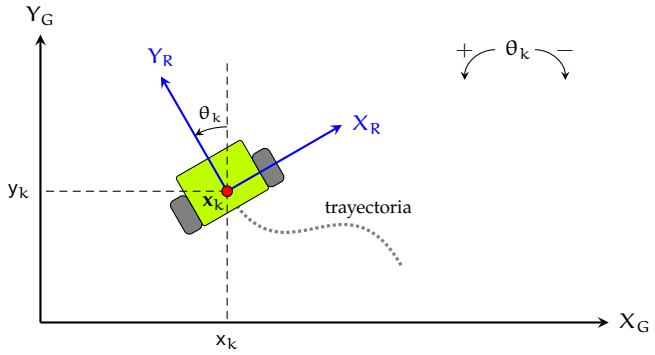


Figura 3.3.1: Relación entre el sistema de referencia local del robot y el sistema de referencia global.

Como punto identificativo (x_k, y_k) del robot, coincidente con el origen del sistema de referencia local, se ha elegido el punto medio del eje de las ruedas motrices, apuntando el eje Y_R en la dirección de avance del robot. La orientación del robot (θ_k) se ha definido como el ángulo formado por el eje Y_R con respecto a Y_G, tomando como positivo el giro en el sentido contrario a las agujas del reloj. El origen del sistema de referencia global se debe elegir arbitrariamente en algún punto del espacio cartesiano.

3.3.3 FUNCIÓN DE TRANSICIÓN DE ESTADOS

En el EKF, la función de transición de estados $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ es una función no lineal que, conociendo la entrada de control $(\mathbf{u}_k)$ que ha modificado el estado del sistema, permite calcular el nuevo estado del sistema $(\mathbf{x}_{k+1})$ a partir del estado actual $(\mathbf{x}_k)$. En el caso de nuestro problema de localización, esta función coincide con el modelo odométrico de nuestro robot diferencial (ver apéndice B), y se utiliza para calcular la nueva posición del robot a partir del desplazamiento relativo medido por sus *encoders*.

El vector $\mathbf{u}_k$ es el resultado de la acción de control medido mediante los *encoders* del robot, y representa el desplazamiento del robot en su

sistema de referencia local al pasar del instante de tiempo k al instante $k+1$:

$$\mathbf{u}_k = \begin{bmatrix} \Delta S_{k,i} \\ \Delta S_{k,d} \end{bmatrix} \tag{3.3.3}$$

donde $\Delta S_{k,i}$ y $\Delta S_{k,d}$ son los desplazamientos realizados por las ruedas izquierda y derecha, respectivamente, y se obtienen directamente a partir de los valores proporcionados al muestrear cada uno de los *encoders*:

$$\begin{aligned} \Delta S_{k,i} &= \frac{\Delta N_i}{n} \cdot C \\ \Delta S_{k,d} &= \frac{\Delta N_d}{n} \cdot C \end{aligned} \tag{3.3.4}$$

Los valores ΔN_i y ΔN_d indican el número de ticks contados por el *encoder* izquierdo o derecho durante el periodo de muestreo, n es la resolución del *encoder* en ticks por vuelta y C es perímetro de las ruedas.

Al transformar los desplazamientos relativos del vector $\mathbf{u}_k$ del sistema de referencia *local* al *global* se obtiene la función de transición de estados:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} x_k - \frac{\Delta S_{k,d} + \Delta S_{k,i}}{2} \cdot \sin\left(\theta_k + \frac{\Delta S_{k,d} - \Delta S_{k,i}}{2b}\right) \\ y_k + \frac{\Delta S_{k,d} + \Delta S_{k,i}}{2} \cdot \cos\left(\theta_k + \frac{\Delta S_{k,d} - \Delta S_{k,i}}{2b}\right) \\ \theta_k + \frac{\Delta S_{k,d} - \Delta S_{k,i}}{b} \end{bmatrix} \tag{3.3.5}$$

donde b es la distancia entre las ruedas motrices del robot. La obtención de estos resultados se encuentra completamente detallada en el apéndice B, y se puede expresar de forma más sencilla en función de los términos ΔS_k y $\Delta\theta_k$, obteniéndose:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} -\Delta S_k \cdot \sin\left(\theta_k + \frac{\Delta\theta_k}{2}\right) \\ \Delta S_k \cdot \cos\left(\theta_k + \frac{\Delta\theta_k}{2}\right) \\ \Delta\theta_k \end{bmatrix} \tag{3.3.6}$$

Los términos ΔS_k y $\Delta\theta_k$ representan el desplazamiento y cambio de orientación del robot en su sistema de referencia *local*. Ambos se pueden calcular mediante las expresiones:

$$\Delta S_k = \frac{\Delta S_{k,d} + \Delta S_{k,i}}{2} \tag{3.3.7}$$

$$\Delta\theta_k = \frac{\Delta S_{k,d} - \Delta S_{k,i}}{b} \tag{3.3.8}$$

3.3.4 RUIDO DEL SISTEMA

Existen dos causas principales que introducen errores en los cálculos del modelo odométrico representado en la función de transición de estados:

- Cuando la velocidad angular del robot es demasiado alta en comparación con el periodo de muestreo, el modelo odométrico no es perfecto (ver B.3) y se producen pequeños errores en los cálculos. Esto se debe a que el modelo utiliza la aproximación para angulos pequeños $\sin(\theta) \approx \theta$.

- Los *encoders*, al igual que cualquier otro sensor, proporcionan medidas que están sujetas a errores. Así, al calcular la distancia que se ha desplazado cada rueda (3.3.3), el resultado obtenido es una aproximación del valor real.

Estas dos fuentes de ruido se interpretan en conjunto como la incertidumbre que afecta a la distancia que se ha desplazado cada rueda durante el periodo de muestreo. Dicha incertidumbre se ha decidido modelar como un valor proporcional al valor absoluto de la distancia recorrida por cada rueda, de modo que la matriz de covarianza del sistema se expresa como:

$$\mathbf{Q}_k = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_i^2 \end{bmatrix} \tag{3.3.9}$$

donde

$$\begin{aligned} \sigma_d &= k_d \cdot |\Delta S_{k,d}| \\ \sigma_i &= k_i \cdot |\Delta S_{k,i}| \end{aligned} \tag{3.3.10}$$

y los términos $\Delta S_{k,d}$ y $\Delta S_{k,i}$ son las distancias recorridas por las ruedas derecha e izquierda, respectivamente, mientras que k_d y k_i son constantes que recogen la incertidumbre producida por las dos fuentes de error citadas. Los valores para las constantes de error k_d y k_i dependen del robot y del entorno y deben seleccionarse experimentalmente estudiando diferentes movimientos del robot. Se ha supuesto que el ruido que afecta al sistema es ruido blanco gaussiano, de modo que los errores en cada una de las ruedas son independientes, y por ello los elementos que no están en la diagonal de la matriz de covarianza son cero.

CAPÍTULO 3.4

MODELO DE OBSERVACIÓN

En el capítulo previo se ha descrito el modelo odométrico del sistema, cuya función consiste en estimar la posición del robot a partir de las medidas propioceptivas proporcionadas por los *encoders* instalados en las ruedas motrices del robot. A pesar de que la odometría ofrece buenas estimaciones a muy corto plazo, es bien sabido a más largo plazo el error en la estimación de la posición crece ilimitadamente [16]. Por ello, en el EKF aplicado a la localización, normalmente se emplean medidas propioceptivas en la fase de predicción, mientras que la fase de corrección se basa en un modelo de observación que corrige los errores acumulativos del modelo odométrico por medio de medidas exteroceptivas.

3.4.1 SENSOR VIRTUAL

Cualquier robot que pretenda obtener algún tipo de información externa debe está equipado con un conjunto de sensores exteroceptivos que le permitan recabar datos de su entorno. Algunos sensores como las radio balizas o los tags RFID ofrecen directamente una medida de la posición y de la orientación de las características que se desean observar. El principal inconveniente de estos sistemas es que requieren modificar cualquier entorno en el que se vaya a trabajar mediante la ubicación de emisores o receptores en puntos estratégicos.

Otros sensores, como los de visión o de distancia no necesitan alterar el entorno, pero a cambio los algoritmos para calcular la ubicación de las características a observar, son más complejos. Dentro de esta opción, existe la posibilidad de crear un *sensor virtual*, es decir, un sensor que obtiene sus observaciones a partir de la fusión de los datos procedentes de varios dispositivos.

En este trabajo las observaciones se efectúan mediante un sensor virtual que corresponde a la fusión sensorial de las imágenes procedentes de una cámara de visión junto con los datos de distancia obtenidos por un láser de medición de distancias. De este modo se logra dotar de profundidad a los datos visuales sin necesidad de recurrir a otros métodos como la visión estéreo o escáneres de datos tridimensionales, que requieren mucha mayor potencia de cálculo.

Nuestro sensor virtual está diseñado para detectar puertas en su rango de visión (ver capítulos 5.2 y 5.3), de modo que cada vez que una puerta es observada, el sensor devuelve la ubicación del centroide de la puerta en el sistema de referencia local del robot:

$$\mathbf{z}^r = \begin{bmatrix} P_x^r \\ P_y^r \end{bmatrix} \tag{3.4.1}$$

3.4.2 ESPECIFICACIÓN DEL MODELO DE OBSERVACIÓN

De acuerdo a lo expuesto en el apartado 3.1.1, el modelo de observación en un EKF debe responder a la ecuación:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k \tag{3.4.2}$$

Sin embargo, aplicado al problema de localización, el modelo debe modificarse ligeramente, pasando a expresarse como:

$$\mathbf{z}_k^r = \mathbf{h}(\mathbf{x}_k, \mathbf{z}_k^g) + \mathbf{w}_k \tag{3.4.3}$$

El significado de esta ecuación es que si el robot se encuentra en la posición determinada por el estado $\mathbf{x}_k$, entonces el modelo predice la ubicación de las características a observar desde el punto de vista del robot ($\mathbf{z}_k^r$) a partir de las características en el sistema de referencia global ($\mathbf{z}_k^g$). Además, cada característica se ve afectada por el modelo de ruido $\mathbf{w}_k$.

En nuestro caso, las características son las puertas detectadas por el sensor virtual de nuestro robot, y sus coordenadas se encuentran almacenadas en un SIG accesible mediante consultas a una base de datos espacial (ver capítulo 5.4). En el SIG, las coordenadas de las puertas están almacenadas en coordenadas globales:

$$\mathbf{z}^g = \begin{bmatrix} P_x^g \\ P_y^g \end{bmatrix} \tag{3.4.4}$$

y mediante la función de observación $\mathbf{h}(\cdot, \cdot)$ se convierten a coordenadas robocéntricas. Esta transformación de coordenadas es fundamental en el EKF, y sirve para comparar las observaciones con las predicciones de las observaciones. En nuestro caso, se compara la ubicación de las puertas observadas por el sensor virtual del robot, con la posición que deberían ocupar las puertas de acuerdo al SIG. Las diferencias entre las observaciones y las predicciones permiten al EKF refinar la posición del robot estimada por el modelo del sistema.

3.4.3 FUNCIÓN DE OBSERVACIÓN

El objetivo de la función de observación es transformar las coordenadas de posición de una puerta en el sistema de referencia global ($\mathbf{z}^g$) a coordenadas en el sistema de referencia local del robot ($\mathbf{z}^r$). Las ecuaciones de esta transformación se pueden averiguar trasladando y rotando el sistema de referencia global hasta que se superponga con el local, como se ilustra en la figura 3.4.1. En la parte izquierda de la figura se muestra una puerta en coordenadas globales, mientras que en la parte derecha se han añadido además las coordenadas desde un punto de vista robocéntrico.

Si se traslada el sistema de referencia global, $\{X_G, Y_G\}$, a la posición en la que se encuentra el robot, se obtiene el sistema de referencia $\{X'_G, Y'_G\}$. Utilizando coordenadas homogéneas, esta traslación se puede expresar como una multiplicación de matrices. Así, $\mathbf{z}^g$ en el sistema $\{X'_G, Y'_G\}$ se expresa mediante la ecuación (3.4.5). A continuación, rotando el sistema $\{X'_G, Y'_G\}$ un ángulo θ_k se obtiene $\mathbf{z}^g$ en el sistema $\{X_R, Y_R\}$, tal y como se indica en la ecuación (3.4.6). Finalmente, seleccionando únicamente

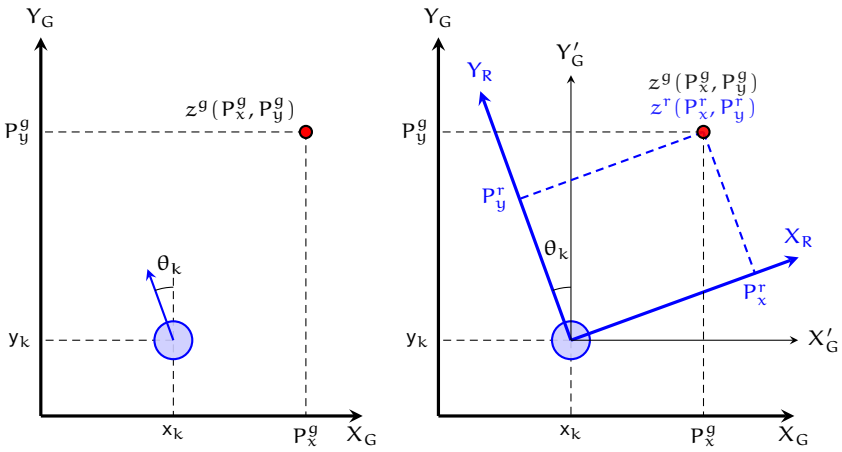


Figura 3.4.1: Transformación de coordenadas del sistema de referencia global al sistema de referencia del robot.

las funciones coordenadas cartesianas y resolviendo la multiplicación, se obtiene la función de observación descrita en la ecuación (3.4.7).

$$\begin{bmatrix} 1 & 0 & -x_k \\ 0 & 1 & -y_k \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_x^g \\ P_y^g \\ 1 \end{bmatrix} \tag{3.4.5}$$

$$\begin{bmatrix} P_x^r \\ P_y^r \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_k) & \sin(\theta_k) & 0 \\ -\sin(\theta_k) & \cos(\theta_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_k \\ 0 & 1 & -y_k \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_x^g \\ P_y^g \\ 1 \end{bmatrix} \tag{3.4.6}$$

$$\mathbf{h}(\mathbf{x}_k, \mathbf{z}^g) = \begin{bmatrix} P_x^r \\ P_y^r \end{bmatrix} = \begin{bmatrix} (P_x^g - x_k) \cdot \cos(\theta_k) + (P_y^g - y_k) \cdot \sin(\theta_k) \\ -(P_x^g - x_k) \cdot \sin(\theta_k) + (P_y^g - y_k) \cdot \cos(\theta_k) \end{bmatrix} \tag{3.4.7}$$

3.4.4 RUIDO EN LAS OBSERVACIONES

Cada vez que el *sensor virtual* del robot detecta una puerta, $\mathbf{z}^r$, la identifica mediante las coordenadas cartesianas correspondientes al centroide de la puerta (P_x^r, P_y^r), especificadas en el sistema de referencia local del robot, $\{X_R, Y_R\}$, tal y como se ilustra en la figura 3.4.2. Además de las coordenadas, es necesario cuantificar la incertidumbre en la detección, es decir, se necesita definir un modelo gaussiano de error que le permita al EKF saber la confianza que se tiene en que la posición de la puerta detectada es correcta.

Dado que en nuestro EKF las coordenadas de las puertas están definidas en un sistema de referencia cartesiano, del mismo modo la matriz de covarianza para definir la incertidumbre en la detección de la puerta debe estar especificada en coordenadas cartesianas:

$$C_{xy} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \tag{3.4.8}$$

En la figura 3.4.2, la incertidumbre se representa mediante una elipse de error (ver apéndice F), obtenida a partir de la matriz de covarianza C_{xy}. La incertidumbre se ha modelado teniendo en cuenta dos factores:

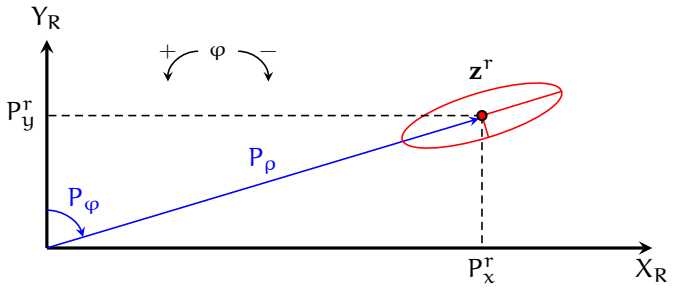


Figura 3.4.2: Modelo de ruido para la observación de una puerta.

1. que sea proporcional a la distancia a la puerta observada, de modo que las puertas más cercanas tengan más peso en la corrección de la posición.
2. que permita establecer un cierto error en la dirección en que la puerta se ha detectado.

Esta decisión se ha tomado teniendo en cuenta que a pesar de que el error en la medida del láser no depende de la distancia al objeto medido (ver apartado 4.1.2.2), hemos comprobado que la tasa de aciertos del algoritmo propuesto para detectar puertas disminuye a medida que las puertas están más lejos del robot. La razón es que las imágenes tomadas por la cámara están en perspectiva, y lógicamente los objetos son de menor tamaño cuanto más lejos están, detectándose con mayor dificultad. Por ello, es más conveniente definir la incertidumbre utilizando coordenadas polares, y posteriormente transformarla a coordenadas cartesianas mediante la ley de propagación de la incertidumbre (ver apéndice E). Otra razón a favor de las coordenadas polares, es que a no ser que el centroide de la puerta esté sobre X_R o Y_R, la elipse de error estará girada con respecto al sistema de referencia $\{X_R, Y_R\}$. Esto significa que los elementos no diagonales C_{xy} no son nulos, y desafortunadamente no se pueden calcular con facilidad.

3.4.4.1 *Incertidumbre en coordenadas polares*

Tomando como referencia para medir el ángulo el eje Y_R y tomando como positivo el giro en el sentido contrario a las agujas del reloj, las coordenadas de la puerta $\mathbf{z}^r$ en formato polar son (P_ρ, P_φ) (ver fig. 3.4.2), cumpliéndose que:

$$\begin{aligned} P_x^r &= -P_\rho \cdot \sin(P_\varphi) \\ P_y^r &= P_\rho \cdot \cos(P_\varphi) \end{aligned} \tag{3.4.9}$$

El error en la distancia de detección de la puerta, σ_ρ, es proporcional a la distancia y se corresponde en la imagen 3.4.2 como el semieje mayor de la elipse. Asimismo, el error en el ángulo de detección, σ_φ se corresponde con el semieje menor de la elipse. Esto nos da la siguiente matriz de covarianza:

$$C_{\rho\varphi} = \begin{bmatrix} \rho \cdot \sigma_\rho^2 & 0 \\ 0 & \sigma_\varphi^2 \end{bmatrix} \tag{3.4.10}$$

donde los términos no diagonales son cero, porque se supone que los errores en la distancia y el ángulo son independientes.

3.4.4.2 *Propagación del error*

Para obtener la matriz de covarianza en coordenadas cartesianas (C_{xy}) a partir del modelo de error en coordenadas polares ($C_{\rho\varphi}$), hay que propagar el error utilizando la ley de propagación de la incertidumbre:

$$\mathbf{C}_{xy} = \nabla\mathbf{f} \cdot C_{\rho\varphi} \cdot \nabla\mathbf{f}^{T} \tag{3.4.11}$$

donde $\nabla\mathbf{f}$ es el jacobiano de la función que define la transformación de las coordenadas polares a cartesianas. Esta función se corresponde con la ecuación (3.4.9). Para simplificar las siguientes expresiones, se cambiarán los nombres de variable P_x^r, P_y^r, P_ρ, P_φ por x, y, ρ, φ, respectivamente. Una vez hecho esto, $\nabla\mathbf{f}$ se calcula como:

$$\nabla\mathbf{f} = \begin{bmatrix} \dfrac{\partial x}{\partial \rho} & \dfrac{\partial x}{\partial \varphi} \\ \dfrac{\partial y}{\partial \rho} & \dfrac{\partial y}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} -\sin(\varphi) & -\rho\cdot\cos(\varphi) \\ \cos(\varphi) & -\rho\cdot\sin(\varphi) \end{bmatrix} \tag{3.4.12}$$

Finalmente, la incertidumbre para una puerta en coordenadas cartesianas se puede obtener en función de las coordenadas polares de la puerta, y en función de los errores σ_ρ y σ_φ como:

$$\mathbf{C}_{xy} = \begin{bmatrix} \rho\sigma_\rho^2 \sin^2(\varphi) + \rho^2\sigma_\varphi^2 \cos^2(\varphi) & (\rho\sigma_\varphi^2 - \sigma_\rho^2)\rho\sin(\varphi)\cos(\varphi) \\ (\rho\sigma_\varphi^2 - \sigma_\rho^2)\rho\sin(\varphi)\cos(\varphi) & \rho\sigma_\rho^2\cos^2(\varphi) + \rho^2\sigma_\varphi^2\sin^2(\varphi) \end{bmatrix} \tag{3.4.13}$$

CAPÍTULO 3.5

RESUMEN

En esta parte se han estudiado los fundamentos del filtro de Kalman, un algoritmo recursivo que permite estimar el estado de un sistema estocástico basándose en observaciones periódicas del estado de dicho sistema. En el capítulo 3.1 se describe el funcionamiento del filtro de Kalman lineal, que se basa en la utilización de dos modelos estadísticos: el modelo del sistema y el modelo de observación. El primero describe la evolución en el tiempo del estado del sistema, mientras que el segundo relaciona las observaciones con el estado. Un aspecto fundamental del KF es que supone que el ruido que afecta al sistema estocástico se puede definir mediante variables aleatorias multivariantes e independientes entre sí que modelan ruido blanco gaussiano.

Los problemas de localización en robótica móvil contienen no linealidades, de modo que necesitan linealizarse mediante el Filtro Extendido de Kalman, descrito en el capítulo 3.2. Su funcionamiento es similar al filtro de Kalman lineal, con la diferencia de que tanto el modelo del sistema como el modelo de observación se linealizan por medio de una serie de Taylor de primer orden. Uno de los inconvenientes de esta linealización es que se introducen pequeños errores en los cálculos que pueden llevar a la divergencia del filtro, pero que pueden solucionarse mediante métodos heurísticos.

Para estimar la posición de un robot mediante un EKF es necesario modelar cuidadosamente el modelo del sistema y el de observación. Así, en el capítulo 3.3 se especifica el estado del robot (su posición y orientación en el plano cartesiano), así como el modelo del sistema, que se basa en la odometría proporcionada por sus *encoders*, y el modelo de ruido que afecta a las mediciones de los *encoders*. A continuación, en el capítulo 3.4 se especifica el modelo de observación, que se utiliza para relacionar las observaciones realizadas por el sensor virtual (visión+láser) del robot con el estado del robot. Además, se define el modelo de ruido que afecta a las observaciones y que ha resultado ser uno de los aspectos más delicados del filtro.

Una vez definidos el modelo del sistema y el modelo de observación, el ciclo predicción-corrección del EKF sirve como herramienta para fusionar la información odométrica con las observaciones realizadas por el sensor virtual del robot. La fase de predicción utiliza el modelo del sistema para efectuar una estimación básica del estado, que se refina en la fase de corrección, siempre que haya observaciones disponibles, y utilizando para ello el modelo de observación.

Parte 4

MODELADO Y CALIBRACIÓN DE SENSORES

CAPÍTULO 4.1

MODELADO DE SENSORES

En este capítulo se estudian y modelan los sensores utilizados en nuestro robot móvil *Morlaco* (ver apéndice A): un LIDAR o escáner láser de medición de distancias, una cámara monocular, y *encoders* rotatorios.

4.1.1 INTRODUCCIÓN

En un mundo estático y determinista, controlar con precisión un robot móvil sería relativamente sencillo puesto que bastaría con obtener un modelo matemático *perfecto* tanto del entorno como de los actuadores del robot. De este modo sería incluso posible, por ejemplo, llevar a un robot de un lugar a otro sin necesitar sensores para obtener información del entorno. Lamentablemente, el mundo real es dinámico y tanto los modelos del entorno como los actuadores están sometidos a error. Esto da lugar a la necesidad de:

1. Utilizar sensores para percibir información del entorno. Así se elimina la limitación de tener que operar en un entorno cuidadosamente controlado, y se abre la posibilidad de trabajar en entornos desconocidos o peligrosos.

2. Emplear modelos de estimación estocásticos, como por ejemplo el filtro de Kalman, como se ha estudiado en la parte 3, para poder manejar la incertidumbre presente en el mundo real.

Al igual que los humanos tenemos diferentes sentidos, los robots se pueden equipar con gran variedad de sensores que hoy en día son capaces de medir estímulos muy variados: color, distancia, fuerza, presión, inclinación, etc. Everett [38] describe y analiza la gran mayoría de sensores existentes. Los sensores se pueden categorizar de diversas maneras. Una de las clasificaciones más habituales consiste en separarlos en *propioceptivos* y *exteroceptivos*. Los del primer tipo sirven para observar el estado interno del robot, como la orientación de un brazo robótico, o el nivel de carga de las baterías. Los sensores exteroceptivos miden estímulos que se originan en el exterior del robot, como distancias a objetos, o la temperatura ambiente, y su función es proporcionar al robot una representación del entorno en el que se encuentra.

Los sensores constituyen la primera etapa del proceso perceptivo. En general, se puede considerar que la percepción engloba el conjunto de pasos necesarios para obtener información de los sensores con el objetivo de conseguir un mayor nivel de abstracción en los datos. Cada sensor está compuesto por uno o más transductores (electromagnéticos, electroacústicos, etc.) que permiten convertir una forma de energía en otra diferente. Por ejemplo, en el caso de una cámara digital, un sensor

CCD o CMOS convierte la energía electromagnética de la luz en un voltaje. El voltaje adquirido está dentro de un rango de valores que permiten interpretarlo como una mayor o menor intensidad de la luz. Una vez que se ha percibido la información, el siguiente paso es aumentar el nivel de abstracción, para lo que se necesitan métodos de extracción de características, como se verá en los capítulos 5.2 y 5.3.

4.1.2 LIDAR

LIDAR es un acrónimo de *LIgth Detection And Ranging* (detección y medición de luz), y es una tecnología óptica para medir la distancia u otras propiedades de un objeto iluminándolo normalmente mediante pulsos de luz láser. Se utiliza entre otros, en campos como la agricultura, geología, topografía, transporte, en el ámbito militar, y por supuesto, también en robótica. Debido al uso de tecnología láser, normalmente se utiliza el término *escáner láser*, o simplemente *láser*, para referirse a un dispositivo LIDAR. Algunas ventajas de este tipo de sensor frente a otros son:

Esta tecnología también se conoce como 'LADAR', acrónimo de 'LAser Detection And Ranging' (detección y medición láser).

- Las medidas se pueden considerar instantáneas. Un sónar, por el contrario, obtiene las medidas con bastante retardo en comparación, y puede necesitar compensaciones por el movimiento del robot.
- Es muy preciso: el error en la medida es del orden de unos pocos milímetros, y la resolución angular está típicamente en el rango $0.25 - 1^{o}$.
- Los datos representan directamente distancias reales a objetos, y el pulso emitido apenas tiene apertura cónica en comparación con un sónar.
- El tiempo de procesamiento para realizar tareas de extracción de características es mucho menor que el necesario para analizar una imagen. Una imagen está compuesta por millones de píxeles, mientras que un láser obtiene unos pocos cientos de medidas.

Algunas desventajas son su elevado precio, y que algunos materiales como el cristal son invisibles para el sensor. Además, los datos se limitan a un plano, si bien esto se puede solventar colocando el sensor en un dispositivo *pan & tilt*. Una alternativa para obtener datos de distancia 3D, y que ha ganado mucha popularidad en competiciones como el DARPA Grand Challenge, son los dispositivos láser comercializados por *Velodyne* [3]. Su modelo más avanzado escanea el entorno de 5 a 15 veces por segundo, con un barrido horizontal de 360^{o} y un barrido vertical de 26.8^{o}, obteniendo 1.3 millones de puntos por segundo. No obstante, su precio es prohibitivo, superando los 70 mil dólares.

4.1.2.1 *Escáner Láser SICK LMS 200*

El modelo que se ha seleccionado para nuestro robot móvil *morlaco* es un SICK LMS 200 (fig. 4.1.1a) . Su funcionamiento se basa en la medida del tiempo de vuelo (fig. 4.1.1b). Para calcular la distancia a un objeto, el láser emite un pulso de luz infrarroja (T). Cuando el pulso incide sobre el objeto más cercano, regresa hacia el sensor (R) y se mide el tiempo

Las siglas 'LMS' corresponden a 'Laser Measurement System' (sistema de medición por láser).

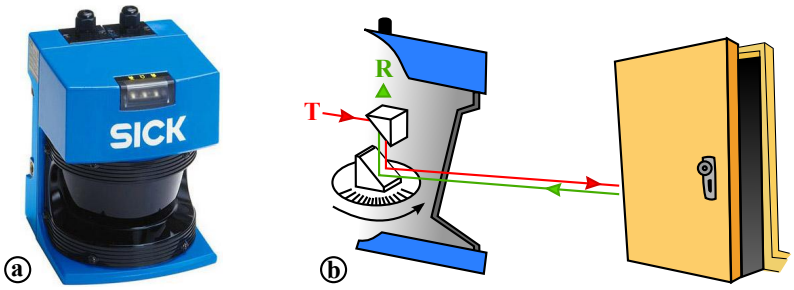


Figura 4.1.1: a) Láser SICK LMS 200. b) Principio de funcionamiento interno.

transcurrido desde la emisión hasta la recepción del pulso (tiempo de vuelo). La distancia se calcula cómo:

$$D = \frac{c \cdot T}{2} \tag{4.1.1}$$

donde c es la velocidad de la luz, y T el tiempo de vuelo. Para obtener distancias en más de una única dirección, se utiliza un espejo rotatorio, obteniendo así un conjunto de medidas que se denomina *barrido láser*. Cada barrido está compuesto por un conjunto de puntos en formato polar:

En inglés se utiliza el término 'scan' o 'laserscan' para referirse a un barrido láser.

$$R = \{(\rho_i, \varphi_i) | i = 1, \ldots, n\} \tag{4.1.2}$$

donde el número de puntos n es variable y depende de la resolución angular seleccionada. Los puntos son adquiridos secuencialmente por el escáner en sentido antihorario, abarcando un semicírculo (fig. 4.1.2a) y con una resolución angular configurable. El término ρ_i indica la distancia al obstáculo detectado al emitir el pulso de luz con orientación φ_i (fig 4.1.2b).

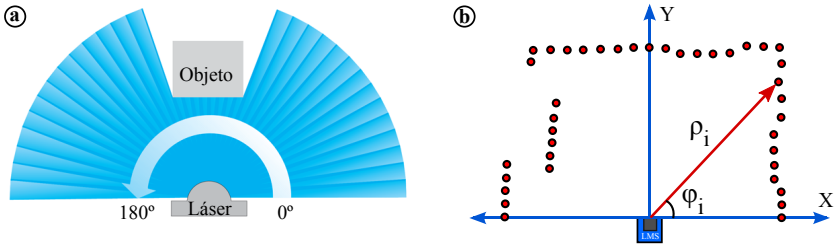


Figura 4.1.2: a) Barrido del SICK LMS 200. b) Coordenadas (ρ, φ) de un punto láser.

Dependiendo de la resolución angular del barrido que se seleccione, la amplitud de barrido, el número de puntos obtenido y el tiempo de muestreo varían de acuerdo tabla 4.1.1. Además de la resolución angular, también se pueden configurar la velocidad de transmisión de datos y la máxima distancia de detección de objetos. En la tabla 4.1.2 se resumen las características técnicas del SICK LMS 200.

4.1.2.2 *Modelado experimental del error*

Cada punto 2D medido por el láser debe tener asociada una incertidumbre, dado que cualquier medida en el mundo real está sujeta a errores. Estudiando estadísticamente suficientes medidas del láser se

RESOLUCIÓN ANGULAR	0.25°	0.50°	1.00°
Amplitud de barrido	100°	180°	180°
Número de mediciones	401	361	181
Tiempo de muestreo	53.33 ms	26.66 ms	13.33 ms
Tasa de muestreo	18 Hz	37 Hz	75 Hz

Tabla 4.1.1: Parámetros variables del láser en función de la resolución angular.

Amplitud de barrido	180°
Resolución angular	0.25°, 0.50°, 1.00°
Tiempo de respuesta	$13 - 53$ ms
Alcance del láser	8, 16, 32, 80 m
Error estadístico (1σ)	5 mm para $d \leqslant 8m$
Interfaz de datos	RS-422 o RS-232
Velocidad de transmisión	9.6, 19.2, 38.4 ó 500 KBaudios
Longitud de onda	Infrarrojo ($\lambda = 905$ nm)
Suministro de voltaje	24 V DC, 1.8 A
Consumo de potencia	20 W
Peso	4.5 kg
Dimensiones	$210 \times 155 \times 156mm$

Tabla 4.1.2: Especificaciones del láser SICK LMS 200 a bordo del robot.

puede obtener una estimación del error que se comete en la medida. Para cada punto se pueden considerar dos fuentes de error: el error en la distancia (ρ), y el error en la dirección de emisión del pulso (φ).

A continuación se detalla el proceso para estimar la varianza del error en la distancia medida (σ_ρ^2), así como la varianza del error en el posicionamiento angular (σ_φ^2). Se asume que el plano de escaneado del láser es paralelo al plano de movimiento del robot a una determinada altura h. El análisis subsiguiente se basa en el procedimiento descrito por Castellanos y Tardós [26].

Error en la distancia medida

Durante los experimentos de calibración se colocó un objeto en frente del robot (fig. 4.1.3a) en un rango de distancias $d_j \in [0.2, 8]m$. Cada distancia d_j se midió N_j veces, seleccionando $N_j = 100$ en todos los casos al ser un número suficientemente significativo como para estimar la varianza. Las medidas de distancia se etiquetaron como $\rho_{m,j}$, con $m \in 1, \ldots, N_j$. La media y la varianza de dichas lecturas se calculó mediante las expresiones (4.1.3) y (4.1.4).

$$\rho_j = \frac{1}{N_j} \sum_{m=1}^{N_j} \rho_{m,j} \tag{4.1.3}$$

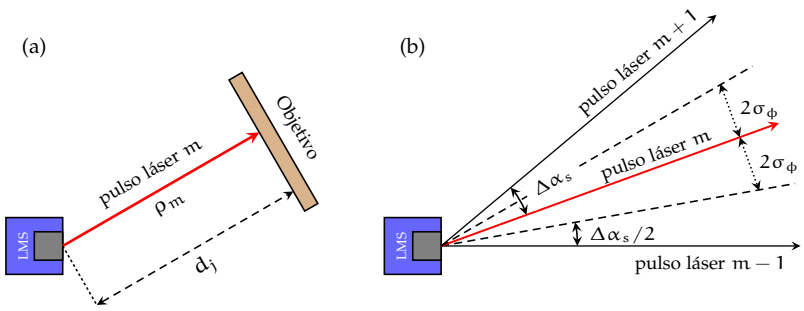


Figura 4.1.3: Calibración de los errores del láser: a) Error en la medida de la distancia y b) Error en el posicionamiento angular.

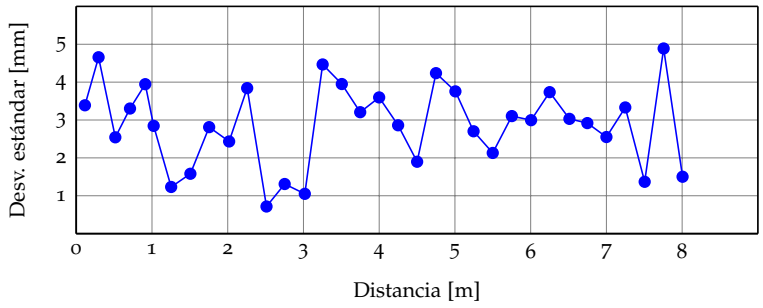


Figura 4.1.4: Medidas de distancia tomadas durante la calibración. Cada punto representa el valor medio de N_j medidas.

$$\sigma^2_{\rho_j} = \frac{1}{N_j - 1} \sum_{m=1}^{N_j} (\rho_{m,j} - \rho_j)^2 \tag{4.1.4}$$

La desviación estándar de las medidas tomadas σ_{ρ_j} se muestra en el gráfico de la figura 4.1.4. El conjunto de valores no parece ajustarse a una curva, sino que más bien varía aleatoriamente entre $1mm$ y $5mm$. Esto coincide con el error estadístico que indica el fabricante en la descripción del sensor (ver tabla 4.1.2). Por ello se ha tomado la cota superior $\sigma_\rho = 5mm$ como base para calcular la varianza, dando como resultado $\sigma^2_\rho = 25mm^2$.

Error en el posicionamiento angular

La segunda fuente de error en el láser es la precisión de la dirección en el que se emite el pulso de luz. Dado que no se ha llegado a un método que permita calibrar este error, se ha asumido que los errores angulares se pueden representar mediante una variable aleatoria con una distribución normal, cuyo margen de error del 95 % viene dado por la resolución angular del sensor. Por lo tanto, considerando que el láser escanea el entorno con una resolución angular $\Delta\alpha_s$, a partir de la (fig. 4.1.3b) se puede obtener:

$$2\sigma_\Phi = \frac{\Delta\alpha_s}{2} \Rightarrow \sigma_\Phi = \frac{\Delta\alpha_s}{4} \tag{4.1.5}$$

Teniendo en cuenta la ecuación anterior, y sabiendo que el láser a bordo del robot se ha configurado con $\Delta\alpha_s = 0.5°$, se obtiene que el error angular puede ser modelado por $\sigma_\Phi = 0.125°$.

4.1.3 CÁMARA

Durante el renacimiento el principal uso de la cámara oscura era realizar pinturas de exteriores respetando las bases de la perspectiva cónica, descubiertas por Filippo Brunelleschi.

Una cámara es un dispositivo que captura la intensidad de la luz procedente de un área determinada del entorno. El término *cámara* procede del latín *camera obscura*, que hasta el siglo XVI consistía simplemente en una habitación oscura en la que sólo se permitía entrar la luz por un pequeño agujero a través de una puerta o una persiana. De este modo la luz entraba en la habitación a través del agujero y se obtenía una imagen invertida del mundo exterior proyectada en una pantalla o una pared opuesta al agujero, como ilustra el grabado de la fig. 4.1.5.

Figura 4.1.5: Ilustración de una cámara oscura en un manuscrito italiano de técnicas militares del siglo XVII.

Las imágenes así obtenidas eran muy poco luminosas y por ello, a mediados del siglo XVI se comenzó a añadir una lente para obtener una mayor luminosidad, al tiempo que se fueron construyendo cámaras portátiles y de menor tamaño. A principios del siglo XVIII, *Joseph Nicéphore Niépce* añadió una placa fotosensible para poder almacenar la imagen, dando lugar a la cámara fotográfica. Más recientemente, en el último cuarto del siglo XX se desarrollaron las primeras cámaras digitales, donde las imágenes se almacenan digitalmente por medio de un sensor CCD (dispositivo de carga acoplada) o un sensor CMOS (semiconductor de óxido metálico complementario).

4.1.3.1 *Webcam 9000 pro de Logitech*

A bordo de nuestro robot *Morlaco* se ha optado por instalar una cámara monocular de *Logitech*, concretamente el modelo *Webcam Pro 9000*. Sus principales características se especifican en la tabla 4.1.3. Durante las pruebas experimentales no se ha utilizado la resolución máxima al proporcionar una tasa de imágenes por segundo lo suficientemente rápida.

Las webcam actuales constan de un sensor CCD/CMOS y un conjunto óptico formado por un diafragma y una lente convergente. En la figura 4.1.6 se muestra el sensor CMOS y el conjunto óptico de la cámara utilizada. El diafragma gradúa la cantidad de luz que entra a la cámara,

Resolución máxima	1600×1200 px (5 fps)
Resolución utilizada	640×480 px (30 fps)
Tipo de sensor CMOS	1/3.2″
Dimensiones del sensor	(4.536 mm $\times$ 3.416 mm)
Apertura horizontal	$\approx 63^{o}$
Apertura vertical	$\approx 50^{o}$
Longitud focal	3,7 mm

Tabla 4.1.3: Especificaciones de la cámara Logitech Webcam 9000 Pro.

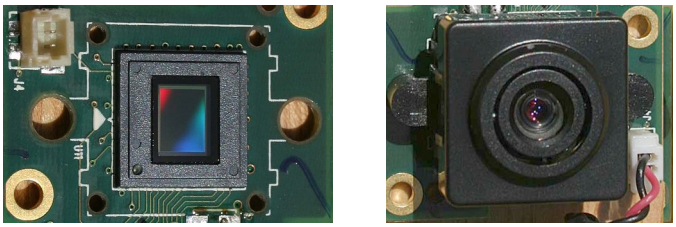


Figura 4.1.6: Sensor CMOS de la webcam 9000 pro de Logitech (izquierda) y su conjunto óptico (derecha).

mientras que la lente hace converger los rayos de luz procedentes del entorno en un punto de la superficie del sensor, que registra la intensidad de la luz recibida.

La imagen almacenada en el sensor es una proyección en dos dimensiones de un mundo en tres dimensiones, y por ello se necesita un modelo matemático que describa qué posición ocupa en la imagen 2D cada punto en 3D del mundo real. El modelo más próximo al de una webcam sería el modelo de lente delgada, si bien es más comúnmente usado el modelo de cámara estenopéica debido a que se puede considerar equivalente al primero, como se explicará a continuación.

4.1.3.2 *Modelo de lente delgada*

Los modelos de lentes pueden llegar a ser bastante complejos, especialmente en el caso de los sistemas formados por varias lentes. En las webcam actuales es habitual encontrar una única lente. En este apartado se estudia el sistema más simple basado en una lente, conocido como modelo de lente delgada.

Una lente delgada es aquella en la que su grosor se considera despreciable en comparación con su longitud focal. Utilizar un modelo de lente delgada sirve para ignorar los efectos ópticos debidos al grosor de la lente y simplifica los cálculos en el trazado de rayos. Según este modelo, los infinitos rayos que emergen de un punto de una escena (O), atraviesan la lente y convergen en un único punto (O') al otro lado de la lente, tal y como se ilustra en la fig. 4.1.7.

Una lente delgada tiene dos puntos focales: el *foco objeto* (F) y el *foco imagen* (F'), ubicándose ambos a la misma *distancia focal* (f) del *centro óptico* de la lente (C). La apertura del diafragma permite controlar la cantidad de luz que entra en la cámara. En este modelo se cumple que todo rayo de luz que pasa por F se refracta paralelo al eje óptico una

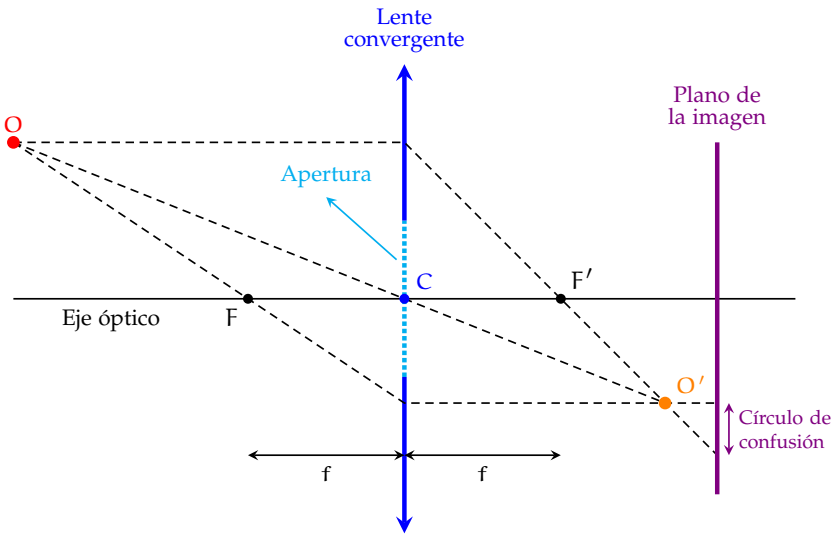


Figura 4.1.7: Modelo de lente delgada, donde se ilustra el centro óptico (C), la distancia focal (f), el foco objeto (F), el foco imagen (F′), el objeto (O) y su imagen (O′).

vez que atraviesa la lente. El foco imagen tiene la propiedad contraria: los rayos que llegan a la lente paralelos al eje óptico pasan por F′ al refractarse en la lente. Los rayos que pasan por el centro óptico de la lente no sufren desviación en su trayectoria. Trazando estos tres rayos para el punto O se obtiene su imagen O′.

En plano de la imagen de una cámara digital está colocado el sensor CCD/CMOS para captar la intensidad de la luz

Para que el punto O′ se visualice nítidamente, tiene que encontrarse proyectado exactamente en el plano de la imagen . De lo contrario, el punto se encuentra desenfocado y en el plano de la imagen no se registra un único punto sino un *círculo de confusión*, dando lugar a una imagen borrosa. Esto se puede arreglar o bien moviendo el plano de la imagen, o desplazando la lente, siendo esta última opción la más habitual. En los primeros tiempos de la fotografía el diseño de las cámaras incluía un fuelle que permitía modificar la distancia entre la lente y el plano de la imagen. Hoy en día esto se logra mediante un sistema denominado *autofoco*, que consiste en un pequeño motor que mueve la lente automáticamente para que el sujeto enfocado quede correctamente proyectado en el plano de la imagen.

4.1.3.3 *Modelo de cámara estenopeica*

El término inglés para 'cámara estenopéica' es 'pinhole camera'.

El modelo de cámara estenopéica describe matemáticamente la proyección de un punto en 3D del mundo real en el plano de la imagen de una cámara estenopéica ideal, donde la apertura del diafragma es infinitesimal, y no se utilizan lentes para enfocar la luz. Esto equivale al modelo de lente delgada siempre y cuando sólo se tuvieran en cuenta los rayos que pasan a través del centro óptico de la lente, puesto que estos rayos son los únicos que no modifican su trayectoria al atravesar el cristal. Al ser prácticamente equivalentes, y debido a su simplicidad, el modelo de cámara estenopeica se utiliza ampliamente en lugar del modelo de cámara de lente delgada.

El funcionamiento de la cámara estenopéica ideal se ilustra en la figura 4.1.8. Como el agujero es infinitesimal, por cada punto del objeto sólo entra en la cámara un único rayo de luz. Los rayos se proyectan en

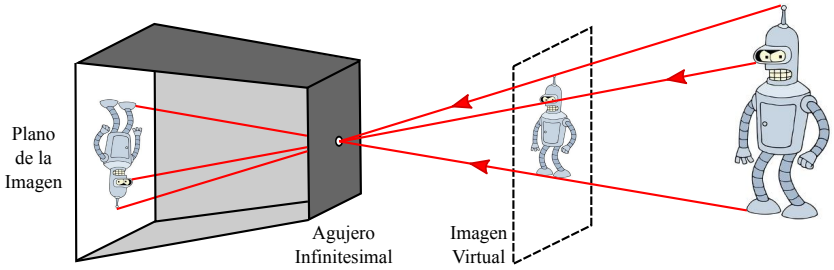


Figura 4.1.8: Funcionamiento de una cámara estenopéica ideal.

la parte posterior de la cámara formando una imagen invertida. Una peculiaridad de esta cámara es que la imagen siempre está enfocada, al no poderse producir círculos de confusión porque sólo hay un rayo por punto. Además, como se verá a continuación, en la descripción del modelo normalmente se sustituye el plano de la imagen por el plano de la imagen virtual, porque así la imagen no está invertida.

El modelo matemático de una camara estenopéica para convertir cada punto del mundo en 3D en un punto de la imagen 2D obedece a una proyección de perspectiva, como se muestra en la figura 4.1.9. El modelo simplemente necesita conocer la ubicación del *centro óptico*

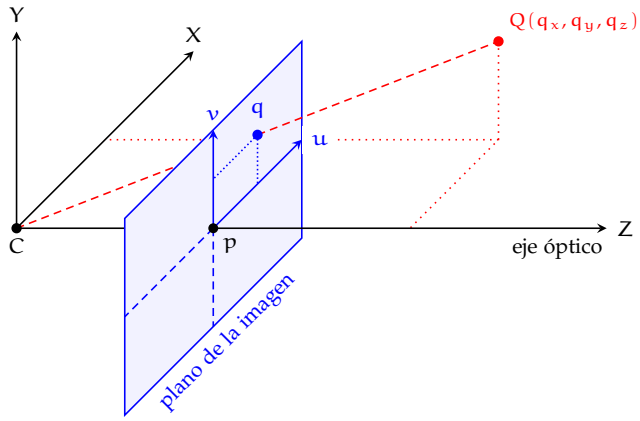


Figura 4.1.9: Modelo de cámara estenopéica: proyección en perspectiva de un punto en 3D en el mundo real (Q) a un punto en 2D (q) en el plano de la imagen.

(o centro de proyección) de la cámara, C, y la ubicación del *plano de la imagen*, que es perpendicular al eje óptico de la cámara. La distancia entre el plano de la imagen y el centro óptico es la *distancia focal*, f. La intersección del plano de la imagen con el eje óptico se denomina punto principal (p) y es el origen del sistema de referencia $\{u, v\}$.

Según este modelo, un punto Q en el espacio con coordenadas (q_x, q_y, q_z) se proyecta en el plano de la imagen en el punto exacto en el que la línea que pasa por Q y por C intersecta el plano de la imagen, dando lugar al punto $q(q_u, q_v)$. Por semejanza de triangulos (ver fig. 4.1.10), se puede calcular fácilmente que las coordenadas (q_x, q_y, q_z) se convierten en $(f \cdot q_x/q_z, f \cdot q_y/q_z, f)$ en el plano de la imagen. Ignorando la última coordenada, se observa que:

$$(q_x, q_y, q_z)^T \mapsto (f \cdot \frac{q_x}{q_z}, f \cdot \frac{q_y}{q_z})^T \tag{4.1.6}$$

es una función de $\mathbb{R}^3$ en $\mathbb{R}^2$ que describe la proyección de un punto 3D en el mundo real a un punto en 2D en el plano de la imagen. Esta proyección resulta imprescindible para efectuar la fusión sensorial de los datos del láser y la cámara de nuestro robot, como se explicará en el capítulo 4.2.

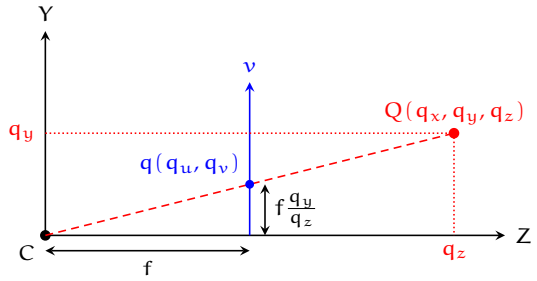


Figura 4.1.10: Proyección de un punto en el plano de la imagen según el modelo de cámara estenopéica.

4.1.4 ENCODERS ROTATORIOS

El término 'encoder' se puede traducir al español como 'codificador'. Sin embargo, en robótica es más común utilizar el anglicismo directamente.

Un *encoder* rotatorio (o de eje) es un dispositivo electromecánico que convierte el movimiento angular de un eje en un código digital. Si el código que se genera para cada posición del eje es único, el *encoder* es *absoluto*, mientras que en otro caso se denomina *relativo* o *incremental*. En robótica móvil se utilizan habitualmente *encoders* incrementales para llevar a cabo tareas odométricas, de tal manera que se puede estimar la velocidad, la distancia recorrida o la posición del robot.

En la fig. 4.1.11 se muestra el esquema general de un *encoder* óptico rotatorio incremental. Está formado por un disco con agujeros o ranuras

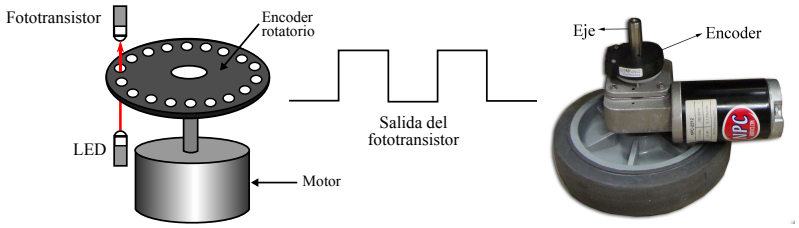


Figura 4.1.11: *Encoder* óptico rotatorio unido a un motor.

cerca de su borde. Se coloca de modo que el eje del disco coincida con el eje del motor y el de la rueda, que normalmente son coincidentes. El número de agujeros en el disco determina la precisión del *encoder*. A un lado del disco se coloca un LED infrarrojo, mientras que al otro lado, y enfrente del LED emisor, se coloca un fototransistor receptor de infrarrojos. Cuando el disco gira, los agujeros hacen que el fototransistor reciba luz de manera intermitente, generándose así una onda cuadrada. Los pulsos detectados por el fototransistor se transforman en distancia con un cálculo basado en el diámetro de la rueda, como se describe en el apéndice B.

Un inconveniente de los *encoders* de un solo canal, como el anterior, es que no pueden detectar la dirección de rotación, y por tanto no sirven para calcular la posición. Para solucionar este problema se usa un segundo canal, desfasado 90º con respecto al primero. Esta técnica

permite determinar qué canal va por delante y averiguar así la dirección de giro, con el beneficio adicional de una mayor resolución. Para nuestro robot hemos adquirido dos *encoders* ópticos incrementales, de *USDigital*, con doble canal, y 2000 pulsos por vuelta. Como los motores seleccionados tienen doble eje, se ha aprovechado el eje interno de cada motor para colocar en ellos los *encoders* (ver fig. 4.1.11 derecha).

CAPÍTULO 4.2

CALIBRACIÓN

El objetivo de este capítulo es obtener un modelo matemático que permita proyectar los datos tomados por el láser en las imágenes obtenidas por la cámara, consiguiendo así añadir información de profundidad a la imagen y efectuar una fusión de datos a bajo nivel. Este proceso se divide en dos fases: calibración intrínseca y calibración extrínseca. Mediante la calibración intrínseca se estiman una serie de parámetros internos de la cámara, mientras que con la calibración extrínseca se estiman los valores relativos de rotación y traslación entre los sistemas de referencia láser-robot y robot-cámara. El procedimiento de calibración que se ha utilizado es el de *Guan et al.* [47].

4.2.1 SISTEMAS DE REFERENCIA

El robot *Morlaco* está equipado con un escáner láser de medición de distancias y una cámara de visión, orientados ambos en la dirección de avance del robot. El sistema de coordenadas del láser es bidimensional y mide distancias en un único plano. Por el contrario, la cámara emplea un sistema de coordenadas tridimensional puesto que obtiene puntos en un espacio 3D, pero que se proyectan en el plano de la imagen, que es bidimensional. Tanto el vehículo como los dos sensores tienen sistemas de referencia independientes, y en la figura 4.2.1 se muestra —en vistas de perfil y planta— la relación existente entre la disposición de los tres sistemas de referencia seleccionados.

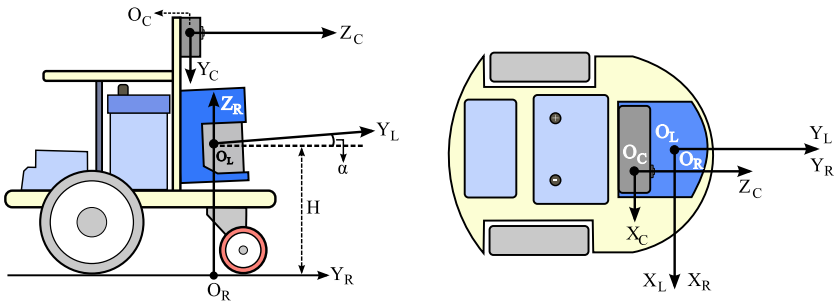


Figura 4.2.1: Sistemas de referencia del robot, láser y cámara.

SISTEMA DE REFERENCIA DE LA CÁMARA: Su origen es O_C, el centro óptico de la cámara. Z_C es su eje óptico, y es perpendicular al plano de la imagen, determinado por los ejes $\{X_C, Y_C\}$.

SISTEMA DE REFERENCIA DEL LÁSER: Su origen es O_L, que coincide con el centro del sistema óptico del láser y está a una altura H del suelo. Los ejes $\{X_L, Y_L\}$ forman un plano prácticamente

paralelo al plano $X_R Y_R$ del sistema de coordenadas del robot. Sin embargo, debido a imperfecciones en la construcción del robot, el plano de medición del láser se encuentra ligeramente inclinado hacia arriba un pequeño ángulo α.

SISTEMA DE REFERENCIA DEL ROBOT: Su origen O_R se ha establecido como la proyección del origen del sistema de referencia del láser (O_L) en el suelo. El eje X_R coincide con X_L; Y_R coincide con la proyección horizontal de Y_L, mientras que Z_R es perpendicular hacia arriba con respecto al plano $X_R - Y_R$.

4.2.2 CALIBRACIÓN DEL LÁSER

El primer paso para poder proyectar los datos medidos por el láser en las imágenes de la cámara es transformar las coordenadas del sistema de referencia del láser al del robot. Para ello hay que estimar la rotación y traslación relativa del láser con respecto al robot mediante un proceso de calibración extrínseca.

En castellano, los ángulos de Tait-Bryan se denominan alabeo (roll), cabeceo (pitch) y guiñada (yaw), aunque en robótica es más habitual utilizar los anglicismos.

En un sistema de coordenadas tridimensional la traslación viene dada por los desplazamientos ΔX, ΔY y ΔZ, mientras que la rotación está determinada por los ángulos de *Tait-Bryan*: *pitch* (R_x), *roll* (R_y) y *yaw* (R_z), ilustrados en la figura 4.2.2a. Las traslaciones en X e Y y la rotación R_z son nulas por la definición de los sistemas de referencia (ver figura 4.2.1). Además, se ha supuesto que la rotación en R_y es despreciable, de modo que los únicos parámetros a estimar son la rotación en R_x, que denominaremos α, así como la altura H a la que está montado con respecto al suelo (traslación en Z).

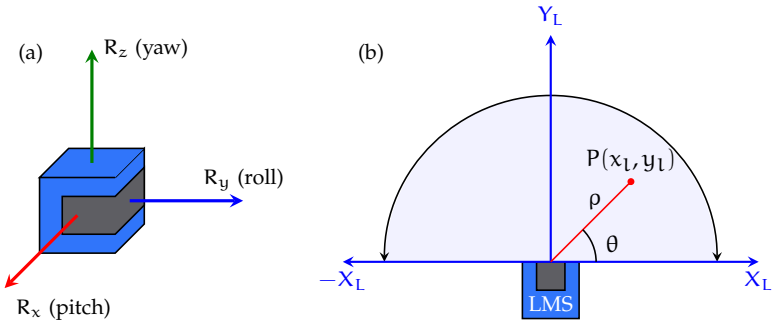


Figura 4.2.2: a) Ángulos de rotación en el láser. b) Detección de un punto en el sistema de referencia del láser.

4.2.2.1 *Modelo matemático de la transformación*

Cada punto medido por el láser está en coordenadas polares, como se ilustra en la figura 4.2.2b. La transformación a coordenadas cartesianas se realiza mediante (4.2.1).

$$\begin{bmatrix} x_l \\ y_l \\ 0 \end{bmatrix} = \begin{bmatrix} \rho \cdot \cos(\theta) \\ \rho \cdot \sin(\theta) \\ 0 \end{bmatrix} \tag{4.2.1}$$

De acuerdo a lo explicado anteriormente, para transformar un punto del sistema de referencia del láser $\{X_L, Y_L, 0\}$ al sistema de referencia del

robot $\{X_R, Y_R, Z_R\}$ se puede hacer mediante la expresión (4.2.2), donde R_L es una matriz de rotación 3×3 alrededor del eje R_x del láser, y T_L es un vector de traslación en Z_R. Desarrollando R_L y T_L, la ecuación anterior se convierte en (4.2.3) y sustituyendo a partir de (4.2.1) se obtiene (4.2.4).

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = R_L \begin{bmatrix} x_l \\ y_l \\ 0 \end{bmatrix} + T_L \tag{4.2.2}$$

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} \tag{4.2.3}$$

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \rho \cdot \cos(\theta) \\ \rho \cdot \sin(\theta) \cdot \cos(\alpha) \\ H - \rho \cdot \sin(\theta) \cdot \sin(\alpha) \end{bmatrix} \tag{4.2.4}$$

4.2.2.2 *Calibración experimental del láser*

Para estimar los parámetros α y H se ha empleado un tablero de calibración con forma de triángulo rectángulo isósceles, representado en la figura 4.2.3 como el triángulo ABC, donde $\angle A = \angle C = 45°$ y $\angle B = 90°$. El plano de escaneado del láser es $O_L DE$, y está ligeramente inclinado hacia arriba un ángulo α con respecto al plano $X_R Y_R$. Esta inclinación hace que el láser mida el punto q (para $\theta = 90°$), cuando debería medir el punto p en el caso de que el plano de escaneado fuera perfectamente paralelo al plano $X_R Y_R$.

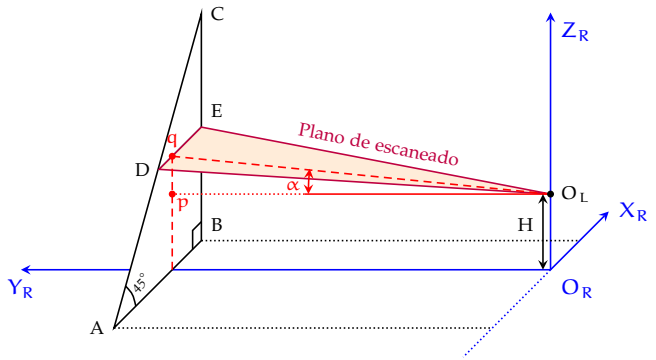


Figura 4.2.3: Calibración de los parámetros del láser: α y H.

A partir de la imagen y según el teorema del coseno se obtiene (4.2.5). Además, $\overline{BC}$ es conocido y $\overline{DE} = \overline{CE}$ por ser un triángulo isósceles, de modo que $Z_{R_E} = \overline{BE} = \overline{BC} - \overline{DE}$.

$$\overline{DE} = \sqrt{\rho_D^2 + \rho_E^2 - 2\rho_D \rho_E \cos(\theta_D - \theta_E)} \tag{4.2.5}$$

Si se coloca el triángulo de calibración a dos distancias distintas, a partir de (4.2.4) se obtiene el sistema de ecuaciones (4.2.6), donde

se conocen todas las variables a excepción de α y H, que se pueden obtener como (4.2.7) y (4.2.8) resolviendo el sistema.

$$\begin{cases} Z_{R_{E1}} = H - \rho_{E1} \cdot \sin(\theta_{E1}) \cdot \sin(\alpha) \\ Z_{R_{E2}} = H - \rho_{E2} \cdot \sin(\theta_{E2}) \cdot \sin(\alpha) \end{cases} \tag{4.2.6}$$

$$\sin(\alpha) = \frac{Z_{R_{E1}} - Z_{R_{E2}}}{\rho_{E2} \cdot \sin(\theta_{E2}) - \rho_{E1} \cdot \sin(\theta_{E1})} \tag{4.2.7}$$

$$H = \frac{Z_{R_{E1}} \cdot \rho_{E2} \cdot \sin(\theta_{E2}) - Z_{R_{E2}} \cdot \rho_{E1} \cdot \sin(\theta_{E1})}{\rho_{E2} \cdot \sin(\theta_{E2}) - \rho_{E1} \cdot \sin(\theta_{E1})} \tag{4.2.8}$$

Por tanto, los parámetros α y H del láser se pueden calcular a partir de las distancias a los puntos en que el plano de medición del láser corta al triángulo (ρ_E, ρ_D) así como los ángulos que corresponden a dichas medidas (θ_E, θ_D), colocando el triángulo a dos distancias distintas. Para lograr una mayor precisión en la calibración se ha colocado el triángulo a 3 distancias distintas y se han efectuado las medidas pertinentes, tal y como se indica en la tabla 4.2.1. A continuación, se han combinado estas pruebas de dos en dos, utilizando las ecuaciones (4.2.7 y 4.2.8), y se han promediado los resultados, obteniendo los valores mostrados en la tabla 4.2.2.

	PRUEBA 1	PRUEBA 2	PRUEBA 3
ρ_E	0.669 m	1.341 m	2.185 m
ρ_D	0.650 m	1.332 m	2.183 m
θ_E	137.00°	145.25°	147.75°
θ_D	158.00°	155.50°	153.75°

Tabla 4.2.1: Parámetros resultantes de la medición del triángulo de calibración.

	PRUEBAS 1-2	PRUEBAS 2-3	PRUEBAS 1-3	PROMEDIO
α	0.40°	1.47°	1.01°	0.96°
H	0.258 m	0.244 m	0.254 m	0.252 m

Tabla 4.2.2: Parámetros resultantes de la calibración del láser.

4.2.3 CALIBRACIÓN DE LA CÁMARA

Para calibrar la cámara se ha utilizado el método de *Roger Y. Tsai* [107], que se basa en el modelo de cámara estenopéica de proyección en perspectiva de 3D a 2D teniendo en cuenta un modelo de distorsión radial de primer orden. A pesar de que las lentes de las cámaras se ven afectadas principalmente por distorsión radial y tangencial, *Tsai* excluye de su modelo la tangencial al considerar que no es necesario tenerla en cuenta en aplicaciones para visión industrial y además causaría inestabilidad numérica en los cálculos.

El algoritmo de calibración propuesto por *Tsai* se divide en dos etapas: en la primera calcula la traslación en los ejes X e Y y la orientación relativa de la cámara en 3D con respecto al sistema de referencia global; en la segunda, calcula la distancia focal, los coeficientes de distorsión de la lente y la traslación en el eje Z. En total el modelo calcula diez parámetros, de los cuales cuatro son intrínsecos y seis extrínsecos.

PARÁMETROS INTRÍNSECOS: sirven para efectuar la transformación de un punto en 3D en el sistema de referencia de la cámara a un punto en 2D en el sistema de coordenadas de la imagen, y su descripción se muestra en la tabla 4.2.3.

PARÁMETROS EXTRÍNSECOS: definen la trasformación de un punto en 3D en el sistema de referencia global a un punto en 3D en el sistema de referencia de la cámara, y se encuentran descritos en la tabla 4.2.4.

PARÁMETRO INTRÍNSECO	DESCRIPCIÓN DEL PARÁMETRO
f	Distancia focal efectiva en el modelo de cámara estenopéica, que equivale a la distancia del origen de coordenadas de la cámara al plano de la imagen.
$\mathbf{K_1}$	Coeficiente de distorsión radial de primer orden.
(**Cx**, **Cy**)	Coordenadas (en píxeles) del centro de distorsión radial de la imagen.

Tabla 4.2.3: Parámetros intrínsecos de la cámara.

PARÁMETRO EXTRÍNSECO	DESCRIPCIÓN DEL PARÁMETRO
$\mathbf{R_x}, \mathbf{R_y}, \mathbf{R_z}$	Ángulos de rotación para la transformación del sistema de referencia global al de la cámara.
$\mathbf{T_x}, \mathbf{T_y}, \mathbf{T_z}$	Componentes de desplazamiento para la transformación del sistema de referencia global al de la cámara.

Tabla 4.2.4: Parámetros extrínsecos de la cámara.

4.2.3.1 *Modelo matemático de la transformación*

En la figura 4.2.4 se ilustra el problema de calibración de la cámara y los distintos elementos a tener en cuenta. La cámara, —representada mediante el sistema de referencia $\{X_C, Y_C, Z_C\}$— puede estar rotada y desplazada en cualquiera de las tres direcciones del espacio con respecto al sistema de referencia del robot —representado por $\{X_R, Y_R, Z_R\}$—. En la imagen se observa además, como un mismo punto en el espacio tiene diferentes coordenadas dependiendo de si se está observando desde el sistema de referencia del robot (P_r) o desde el de la cámara (P_c). Finalmente se puede apreciar también como el punto observado queda registrado en el plano de la imagen $\{X_I, Y_I\}$, que se encuentra a una

distancia focal f del sistema de referencia de la cámara. El plano $\{U, V\}$ es similar al plano de la imagen, pero supone cambiar de unidades de distancia a píxeles basándose en el tamaño del sensor CMOS de la cámara, como se explicará más adelante.

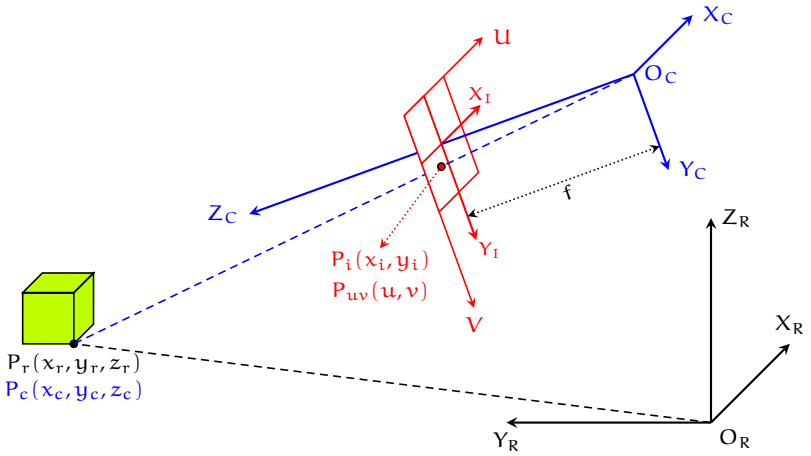


Figura 4.2.4: Observación de un mismo punto P en el sistema de referencia del robot (P_r), el sistema de referencia de la cámara (P_c) y proyección del punto en el plano de la imagen (P_i).

La transformación de un punto en coordenadas del robot (x_r, y_r, z_r) a coordenadas de la cámara (x_c, y_c, z_c) se puede hacer mediante las ecuaciones (4.2.9) – (4.2.11), donde R_C y T_C son la matriz de rotación y el vector de traslación obtenidos al efectuar el proceso de calibración.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R_C \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + T_C \tag{4.2.9}$$

$$R_C = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \qquad T_C = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{4.2.10}$$

$$\begin{aligned}
R_{11} &= \cos(R_y)\cos(R_z) \\
R_{12} &= -\cos(R_x)\sin(R_z) + \sin(R_x)\sin(R_y)\cos(R_z) \\
R_{13} &= \sin(R_x)\sin(R_z) + \cos(R_x)\sin(R_y)\cos(R_z) \\
R_{21} &= \cos(R_y)\sin(R_z) \\
R_{22} &= \cos(R_x)\cos(R_z) + \sin(R_x)\sin(R_y)\sin(R_z) \\
R_{23} &= -\sin(R_x)\cos(R_z) + \cos(R_x)\sin(R_y)\sin(R_z) \\
R_{31} &= -\sin(R_y) \\
R_{32} &= \sin(R_x)\cos(R_y) \\
R_{33} &= \cos(R_x)\cos(R_y)
\end{aligned} \tag{4.2.11}$$

Una vez transformadas las coordenadas del punto al sistema de referencia de la cámara, el siguiente paso es averiguar en qué lugar del plano de la imagen se efectuará la proyección del punto observado. En la figura 4.2.5 se muestra una simplificación de la figura 4.2.4, donde

aparecen únicamente los ejes de la cámara y el plano de la imagen en vista de perfil. Para obtener las coordenadas del punto en el plano de la imagen (x_i, y_i), por semejanza de triángulos se tiene que $f/z_c = y_i/y_c$, y del mismo modo se puede establecer una relación de semejanza para x_i. El resultado se refleja en el sistema de ecuaciones (4.2.12).

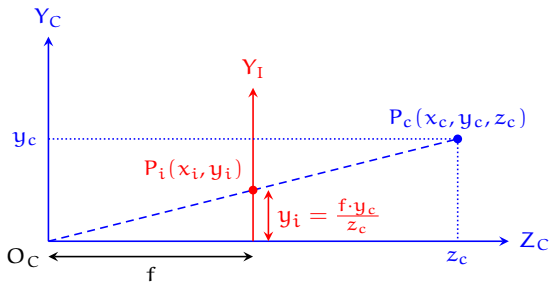


Figura 4.2.5: Proyección de un punto (P_c) en el plano de la imagen (P_i).

$$\left.\begin{aligned} x_i &= f \cdot \frac{x_c}{z_c} \\ y_i &= f \cdot \frac{y_c}{z_c} \end{aligned}\right\} \tag{4.2.12}$$

En una cámara digital, el plano de la imagen se sitúa sobre un sensor (un array de células fotosensibles) que es el encargado de registrar la intensidad de luz procedente de cada punto de la escena observada. Por lo tanto, (4.2.12) expresa las coordenadas (x_i, y_i) en las mismas unidades que (x_c, y_c, z_c), es decir, en unidades de distancia que indican la posición de la superficie del sensor donde se ha registrado el punto. Sin embargo, lo que se desea es conocer las coordenadas del punto en píxeles, y para ello se necesita un factor de conversión de distancia a píxeles, que dependerá del tamaño concreto del sensor de la cámara.

En la figura 4.2.6 se muestra una representación de un sensor CMOS, donde w_s y h_s representan las dimensiones del sensor, y n_x y n_y el número de píxeles.

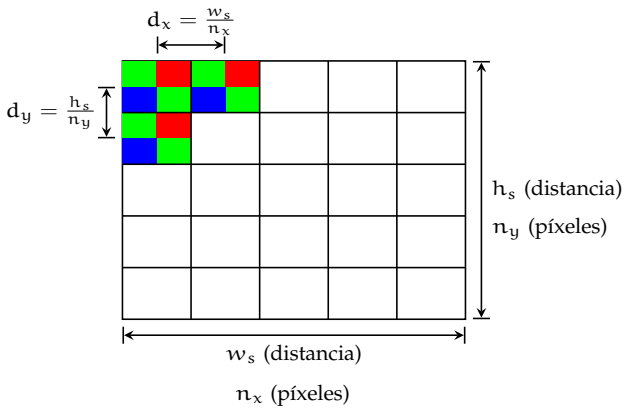


Figura 4.2.6: Dimensiones del sensor CMOS y distancia entre píxeles (d_x, d_y). Cada píxel se forma interpolando dos muestras de color verde, una roja y una azul mediante un filtro de Bayer.

Los factores de conversión $d_x = w_s/n_x$ y $d_y = h_s/n_y$ representan la distancia que existe entre dos píxeles en las dos dimensiones del sensor,

y utilizándolos en la ecuación (4.2.12), se obtienen las coordenadas en píxeles (u, v), como se indica en (4.2.13), donde (C_x, C_y) son las coordenadas del centro de la imagen en píxeles.

$$\left.\begin{aligned} u &= \frac{f \cdot x_c}{d_x \cdot z_c} + C_x \\ v &= \frac{f \cdot y_c}{d_y \cdot z_c} + C_y \end{aligned}\right\} \tag{4.2.13}$$

Combinando los dos pasos anteriores, y utilizando coordenadas homogéneas, la transformación de un punto en coordenadas el robot (x_r, y_r, z_r) a un punto en el plano de la imagen en píxeles (u, v) se resume en las ecuaciones (4.2.14) y (4.2.15).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \frac{f}{d_x} & 0 & C_x \\ 0 & \frac{f}{d_y} & C_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \left[R_c \cdot \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} + T_c \right] \tag{4.2.14}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u/w \\ v/w \\ 1 \end{bmatrix} \tag{4.2.15}$$

4.2.3.2 *Calibración experimental de la cámara*

El algoritmo de calibración de *Tsai* se basa en calcular los parámetros intrínsecos y extrínsecos de la cámara (tablas 4.2.3 y 4.2.4) a partir de un conjunto de puntos con sus coordenadas especificadas tanto en el sistema de referencia del robot (x_r, y_r, z_r), como en el plano de la imagen (u, v).

En la figura 4.2.7 se muestra una de las imágenes utilizadas para la calibración, donde se han identificado manualmente las coordenadas en píxeles de los puntos P_1, P_2 y P_3. Antes de la calibración sería

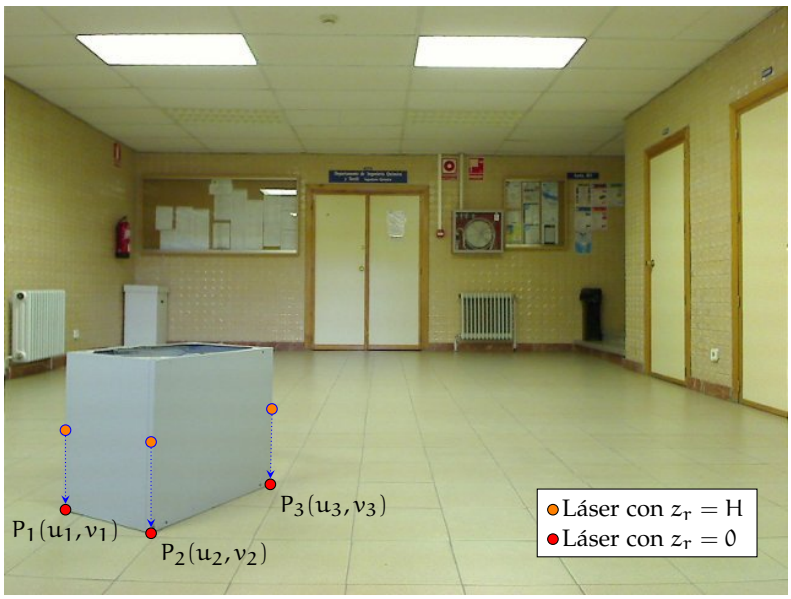


Figura 4.2.7: Una de las imágenes utilizadas para la calibración de la cámara.

complicado averiguar las coordenadas de la imagen correspondientes a los puntos de la caja en los que el láser incidiera ($z_r = H$). Por lo tanto, se han seleccionado los vértices inferiores de la caja ($z_r = 0$) porque es más fácil averiguar sus coordenadas en píxeles en la imagen.

Para obtener las coordenadas correspondientes a los puntos P_{1-3} en el sistema de referencia del robot, se han proyectado las medidas del láser con $z_r = H$ en el suelo, simplemente haciendo $z_r = 0$. Los datos del láser correspondientes a la imagen 4.2.7 se muestran en la figura 4.2.8.

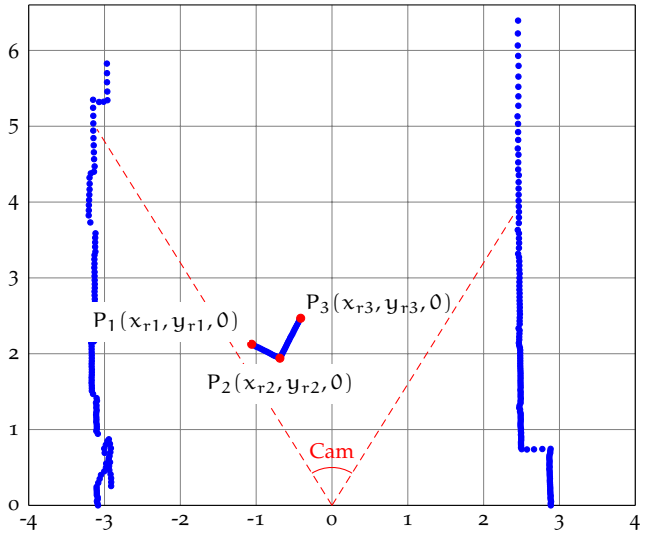


Figura 4.2.8: Captura de datos láser correspondientes a la imagen de la figura 4.2.7.

Para un funcionamiento óptimo del método de calibración de *Tsai*, las coordenadas del conjunto de datos de entrada en el sistema de referencia del robot deben corresponder a puntos distribuidos en el rango de distancias en el que se espera utilizar el modelo. Asimismo, las coordenadas en píxeles correspondientes a los datos de entrada deben abarcar el máximo campo de visión posible de la cámara.

Además, el método de calibración dispone de dos variantes: una para datos ubicados en el mismo plano, y otra para datos en distinto plano, necesitándose en ambos casos un mínimo de 11 puntos para obtener un resultado óptimo. En nuestro caso se ha optado por la calibración de datos coplanarios utilizando un total de 14 datos de entrada, que se recogen en la tabla 4.2.5. Una vez aplicado el método de calibración se han obtenido los parámetros de la tabla 4.2.6.

4.2.4 FUSIÓN DE DATOS

En los apartados previos se han obtenido los parámetros del láser y la cámara, y se han deducido los modelos matemáticos para efectuar transformaciones del sistema de referencia del láser al del robot (4.2.4), y del sistema de referencia del robot al de la cámara (4.2.14). Combinando estas dos ecuaciones, se puede proyectar un punto del láser (ρ, θ) en la imagen $(u, v, 1)$. Así, se logra una fusión de datos a bajo nivel, y se consigue añadir información de profundidad a las imágenes, algo que resulta fundamental para el método de extracción de características que

presentamos en la parte 5. En la figura 4.2.9 se puede ver un ejemplo de la fusión de datos, con los puntos del láser proyectados en la imagen.

Figura 4.2.9: Resultado de la fusión de datos: superposición del láser en la imagen de calibración.

$x_r(m)$	$y_r(m)$	$z_r(m)$	$u(px)$	$v(px)$
-1,902	5,225	0	296	765
-1,554	5,083	0	377	773
-1,302	5,640	0	470	753
-1,060	2,125	0	122	1026
-0,688	1,942	0	298	1073
-0,413	2,469	0	543	974
-0,085	1,957	0	704	1074
0,328	1,962	0	975	1076
-0,052	5,923	0	758	747
0,363	5,941	0	850	747
1,520	4,292	0	1227	814
1,877	4,152	0	1364	822
0,540	2,080	0	1097	1050
0,921	1,889	0	1379	1093

Tabla 4.2.5: Coordenadas para la calibración de la cámara.

	Parámetro	Valor
Intrínsecos	**f**	$3,736\ mm$
	$\mathbf{K_1}$	$3,621 \cdot 10^{-3}\ \frac{1}{mm^2}$
	Cx	$772\ px$
	Cy	$676\ px$

(a)

	Parámetro	Valor
Extrínsecos	$\mathbf{T_x}$	$-34,538\ mm$
	$\mathbf{T_y}$	$-743,826\ mm$
	$\mathbf{T_z}$	$29,232\ mm$
	$\mathbf{R_x}$	$94,07°$
	$\mathbf{R_y}$	$0,31°$
	$\mathbf{R_z}$	$0,45°$

(b)

Tabla 4.2.6: Parámetros resultantes de la calibración.

CAPÍTULO 4.3

RESUMEN

En esta parte se ha ofrecido la información esencial para comprender los fundamentos del método de extracción de características que se describe en la parte 5. En el capítulo 4.1 se estudian los sensores a bordo de nuestro robot *Morlaco*: un escáner láser SICK LMS 200, una cámara monocular de *Logitech* modelo *Webcam Pro 9000* y un par de *encoders* ópticos rotatorios de *USDigital*.

Para el láser se describen sus características y modo de funcionamiento, y a continuación se modela estadísticamente tanto el error que comete en la distancia medida como el error en el posicionamiento angular de cada rayo de luz emitido. En el caso de la cámara, lo fundamental es comprender el modelo matemático correspondiente a la proyección en perspectiva para transformar un punto del mundo real en 3D a un punto 2D en la imagen. Para ello se estudia el modelo de lente delgada, así como una simplificación de este modelo ampliamente empleada y de gran utilidad: el modelo de cámara estenopéica. Finalmente, se estudia brevemente el funcionamiento de los *encoders*.

En el capítulo 4.2 se describe la aplicación de un procedimiento de fusión sensorial a bajo nivel para combinar los datos del láser y la cámara. El procedimiento se basa en definir sistemas de referencia para el láser, la cámara y el robot. A continuación, se calibra el láser, lo que permite estimar su orientación y altura con respecto al robot. La cámara también se calibra, obteniéndose un conjunto de parámetros intrínsecos que definen el modelo de cámara estenopéica, así como un conjunto de parámetros extrínsecos que especifican el desplazamiento y la rotación de la cámara con respecto al robot.

Estos procedimientos de calibración sirven para establecer los modelos matemáticos para transformar un punto del sistema de referencia del láser al del robot, y del sistema de referencia del robot al de la cámara. Finalmente, utilizando las dimensiones y el número de píxeles del sensor CMOS de la cámara, se pueden convertir píxeles en distancia real en metros, efectuando así la transformación final del sistema de referencia de la cámara al plano de la imagen.

Gracias a estas transformaciones, un punto medido por el láser se puede proyectar en su lugar correspondiente en las imágenes obtenidas por la cámara, y también se puede averiguar para un píxel de la imagen su posición en el mundo real. Ambos cálculos son fundamentales para el método de extracción de características que proponemos.

Parte 5

ENFOQUE PROPUESTO

CAPÍTULO 5.1

PERSPECTIVA GENERAL

La inigualable capacidad de comprensión y análisis espacial del ser humano es una de nuestras características más distintivas y nos permite realizar un sinfín de tareas a diario. Adquirir capacidades similares es sin duda una de las características que algún día constituirán la base operativa de cualquier robot. En un intento por avanzar en esta dirección, el objetivo de este trabajo es resolver algunos de los problemas de un robot móvil relacionados con el conocimiento espacial.

El esquema conceptual del sistema que proponemos se muestra en la figura 5.1.1. Como se puede observar los diferentes elementos del

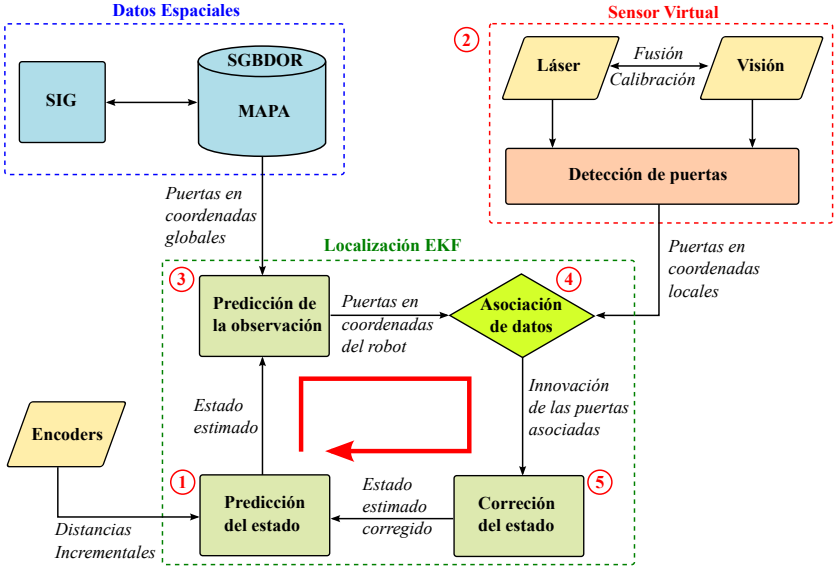


Figura 5.1.1: Sistema de localización basada en un EKF utilizado junto con nuestro sistema de detección de puertas y un SIG para manejar la información espacial del entorno.

esquema se encuentran agrupados en tres rectángulos de línea punteada que pretenden abstraer los aspectos fundamentales que constituyen nuestra propuesta:

- Un novedoso sistema de percepción de información simbólica en el entorno que permite detectar puertas y conocer su posición en el mundo con respecto al robot. Para ello utilizamos un sensor virtual que permite fusionar datos láser y visión monocular.
- Un *Sistema de Información Geográfica* (SIG) para almacenar los datos espaciales del entorno del robot. Gracias a esto el robot puede realizar consultas espaciales al mapa almacenado en la base de

datos espacial subyacente y acceder a la información simbólica del entorno (puertas, ventanas, habitaciones, etc.).

- Un sistema de localización basado en el filtro extendido de Kalman (EKF), integrado junto con nuestro sistema de detección de puertas y el SIG. De este modo el robot se mantiene localizado utilizando la información simbólica que extrae del entorno, contrastándola con los datos espaciales del mapa almacenados en el SGBDOR del SIG.

Este capítulo tiene como objetivo ofrecer una visión global de la solución que planteamos, y en los sucesivos capítulos que componen esta parte se realizará una explicación más detallada. En la sección 5.1.1 se presenta un breve análisis de los beneficios de fusionar datos de un escáner láser y una cámara de visión monocular. Haciendo uso de la fusión sensorial, el apartado 5.1.2 ofrece una visión general del papel que desempeñan láser y visión en el proceso de detección de puertas, y que será analizado con mucho mayor detalle en los capítulos 5.2 y 5.3. La sección 5.1.3 explica concisamente las ventajas de utilizar un SIG en un sistema de localización, y se discute en detalle en el capítulo 5.4. Finalmente, en el apartado 5.1.4 se describe el funcionamiento básico de las dos etapas de un EKF, y se ampliará la discusión en el capítulo 5.5.

5.1.1 FUSIÓN SENSORIAL

Para poder percibir su entorno los robots necesitan sensores exteroceptivos. Entre la enorme variedad de sensores existente [38], dos de los más empleados son las cámaras de visión y el escáner láser, que son precisamente los dos sensores principales de *Morlaco* (ver apéndice A).

Una ventaja de los datos visuales obtenidos por una cámara es que son potencialmente más informativos que los datos de profundidad medidos por sensores de distancia como el láser. Las imágenes permiten detectar objetos en función de su color, forma y tamaño, mientras que la detección de objetos mediante láser es bastante más limitada. Una imagen además obtiene datos del entorno con una resolución angular mucho mayor. Por ejemplo, una cámara con una resolución de 1920×1020 px y un ángulo de visión horizontal de unos 70^{o} tendría una resolución angular de $0,03^{o}$, mientras que un láser trabaja con una resolución angular típica de $0.25 - 1^{o}$. A pesar de sus potenciales ventajas, el principal inconveniente de la visión es que es muy compleja de analizar. La visión es estereoscópica añade la capacidad de percibir profundidad, pero requiere correspondencia de texturas, es computacionalmente costosa y produce resultados imprecisos para muchos píxeles. Además, los entornos interiores a menudo carecen de textura, lo cual es un problema para la visión estéreo.

Por mucho que se analicen las ventajas e inconvenientes de los diferentes sensores existentes, no es recomendable confiar en un único dispositivo para obtener información del entorno, puesto que se pueden producir situaciones en las que las limitaciones de los sensores queden en evidencia. Por ejemplo:

- En un entorno con condiciones de escasa o nula iluminación una cámara de visión quedaría práctica o totalmente inutilizada para cualquier tipo de actividad. Por el contrario, una cámara de

infrarrojos, un escáner láser, o un sónar no se verían afectados por la falta de luz.

- Objetos de cristal, como la mayoría de ventanas y también algunas puertas, resultan invisibles para un escáner láser y muy difícilmente observables por una cámara, pero son fácilmente detectables por un sónar.

Para evitar este tipo de problemas es conveniente utilizar un conjunto de sensores que complementen sus virtudes, mediante lo que se conoce como *fusión sensorial*: combinar datos procedentes de diferentes dispositivos obteniendo una información más robusta y precisa del entorno que cuando los datos se utilizan individualmente.

5.1.1.1 *Beneficios de fusionar láser y visión*

La combinación de sensores de visión con sensores de distancia es especialmente complementaria. De ahí que la fusión sensorial entre láser y visión constituya una parte fundamental del método de detección de puertas que presentamos. Las principales ventajas obtenidas son:

- Cualquier punto del láser o del mundo real puede ser proyectado en una imagen, añadiendo así profundidad a la imagen.
- Se puede realizar extracción de características en el láser y proyectarlas en la imagen. Así se obtiene un resultado final similar a detectar la misma característica en la imagen, pero el coste computacional es mucho menor.
- Un píxel de la imagen puede ser aproximado a un punto del mundo real. Es decir, se consigue añadir profundidad a las imágenes sin necesidad de utilizar visión estéreoscópica. Esto se puede lograr mediante un procedimiento de búsqueda e interpolación, dado que se conoce la equivalencia entre los puntos de un barrido láser y los píxeles correspondientes.
- Es posible resolver el problema de la localización de un robot a partir de la detección de objetos en la imagen (puertas en nuestro caso). Para la localización es fundamental determinar la posición de los objetos detectados en la imagen, que es sencillo de conseguir gracias a la fusión sensorial, pero extremadamente complicado a partir de una única imagen.

En la figura 5.1.2 se muestra la proyección de un barrido láser en la imagen obtenida en el mismo instante de tiempo. Los puntos láser a más de 8 metros de distancia han sido eliminados, puesto que indican que el haz de luz no ha interferido con ningún objeto, y por ello no se muestran puntos entre las dos paredes del pasillo. Este ejemplo se utilizará como hilo conductor en el siguiente apartado para explicar a un alto nivel de abstracción nuestro procedimiento de detección de puertas.

5.1.2 DETECCIÓN DE PUERTAS

El algoritmo de detección de puertas que proponemos se basa en el análisis visual, si bien aprovecha la extracción de características del láser

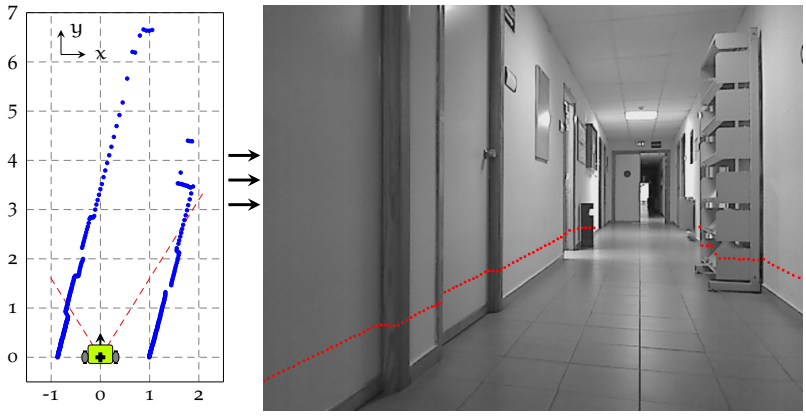


Figura 5.1.2: Izquierda: barrido láser (azul) junto con el ángulo de visión de la cámara (rojo). Derecha: proyección de los puntos láser en la imagen tomada por la cámara en el mismo instante de tiempo.

para calcular regiones en la imagen donde es más probable encontrar puertas. En estas regiones se buscan posibles laterales de las puertas y por último se utilizan criterios métricos y posicionales para completar el proceso de detección. A continuación se explica brevemente este proceso, que será detallado exhaustivamente en los capítulos 5.2 y 5.3.

5.1.2.1 *Regiones de interés*

En entornos interiores, las puertas normalmente están ubicadas en las paredes. Siempre cabe la posibilidad de encontrar alguna puerta en el suelo o en el techo, para acceder a un desván o un sótano, por ejemplo. No obstante, ésta no es la norma, sino una excepción muy poco habitual. Por lo tanto, buscar puertas en partes de la imagen que correspondan al techo o al suelo sería una completa pérdida de tiempo, y es bien sabido que el análisis de una imagen es muy costoso. Por lo tanto, como primer paso se ha decidido definir *regiones de interés* (RDI) en la imagen, restringiendo la búsqueda de puertas a la parte de la imagen donde están las paredes. De este modo no sólo se consigue eliminar de la imagen las áreas en las que es muy poco probable encontrar puertas, o posibles reflejos en el suelo, sino que además el tiempo de procesamiento disminuye al reducirse efectivamente el número de píxeles a analizar.

Detectar el suelo, el techo y las paredes en una imagen es un proceso complejo y costoso. Sin embargo, en un barrido láser las paredes se pueden detectar con un coste computacional mínimo, tal y como proponemos en el capítulo 5.2. Además, gracias a la calibración entre el láser y la cámara que utilizamos (ver capítulo 4.2), se pueden proyectar las paredes detectadas por el láser en la imagen. De esto modo se consigue delimitar en la imagen las regiones de interés donde encontrar puertas, como se muestra en la figura 5.1.3.

5.1.2.2 *Detección de líneas verticales*

La mayoría de las puertas tienen un color o una textura que las hace visualmente diferentes de la pared adyacente. Por esta razón, los píxeles laterales de una puerta normalmente presentan grandes cambios de

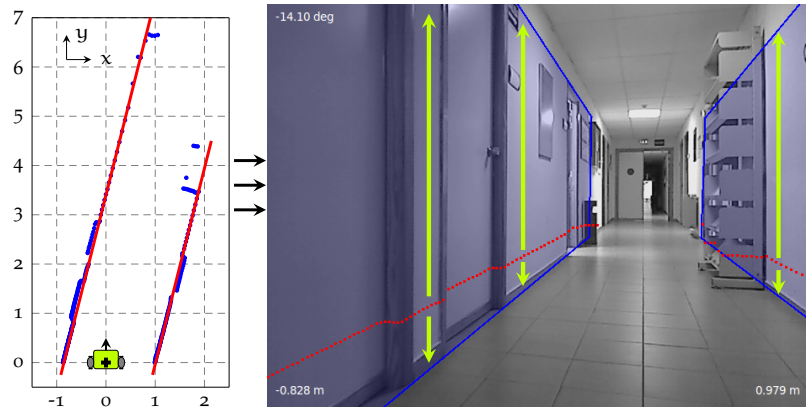


Figura 5.1.3: Izquierda: detección de las paredes (en rojo) en el barrido láser. Derecha, proyección de la línea de la pared a la altura del suelo y a la altura de la puerta, para definir regiones de interés (en azul) en las que buscar puertas.

intensidad con respecto a la intensidad de los píxeles de la pared. Esto da lugar a líneas verticales claramente visibles.

Otro aspecto a tener en cuenta para detectar puertas correctamente, es que dependiendo de la distancia a la que se encuentren, su tamaño en la imagen varía debido a la deformación por perspectiva. Gracias a las RDI definidas anteriormente, se conoce el alto que debe tener una puerta a cualquier distancia, obteniendo así una información muy valiosa y difícil de obtener de otra manera.

Por lo tanto, después de definir las RDI, el siguiente paso que proponemos es detectar líneas verticales en la imagen con el fin de detectar los laterales de las puertas. Para lograrlo, planteamos sumar y restar áreas de píxeles contiguas a lo largo de las RDI, mediante el uso de una ventana deslizante, como se muestra en la figura 5.1.4a. Una de las

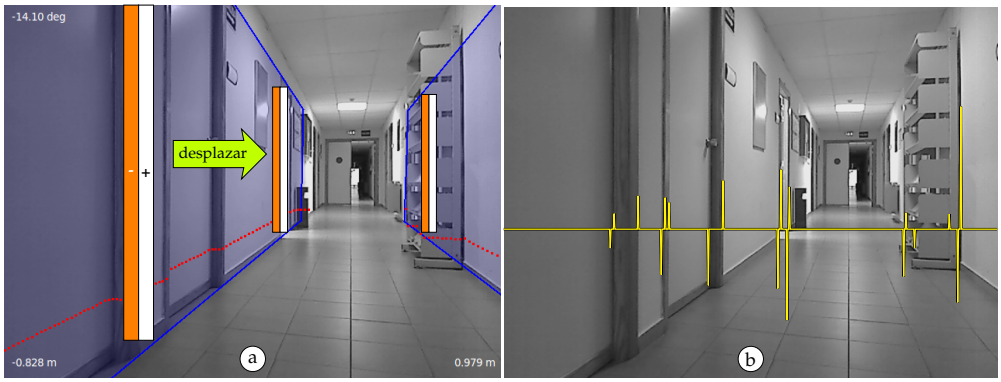


Figura 5.1.4: a) Búsqueda de cambios de intensidad verticales en la imagen utilizando ventanas deslizantes a lo largo de las regiones de interés. b) Resultado de aplicar las ventanas deslizantes a la imagen.

mitades de la ventana suma los píxeles mientras que la otra los resta. Si la ventana se aplica a un área uniforme, como una pared por ejemplo, el resultado es un valor muy cercano a cero. Cuando la ventana pasa justo sobre el lateral de una puerta, los píxeles de una mitad tendrán un valor muy distinto a los de la otra mitad, y como resultado se obtendrá un valor muy distinto de cero. La línea de la figura 5.1.4b muestra el resultado de aplicar la ventana deslizante a la imagen, y se observa

claramente como los picos se corresponden con las líneas verticales. En el capítulo 5.3 se explica en detalle esta técnica y su optimización.

5.1.2.3 *Detección de puertas y cálculo de su posición*

Una vez detectadas las líneas verticales, cada una de ellas constituye un candidato a ser el lateral de una puerta. Es decir, algunas líneas serán realmente el lateral de una puerta, mientras que otras pueden corresponder al lateral de una estantería u otro objeto, como se puede apreciar en la figura 5.1.4b. Por lo tanto, el último paso consiste en buscar qué pares de líneas constituyen una puerta, utilizando criterios de distancia métrica como se explica en el capítulo 5.3.

Además, una vez detectada una puerta, se calcula su posición en el mundo real para poderla utilizar en el proceso de localización del robot. En la figura 5.1.5 se muestra mediante áreas en verde el resultado final de la detección de puertas para el ejemplo utilizado a lo largo de este apartado.

Figura 5.1.5: Resultado final de la detección de puertas.

5.1.3 MANEJO DE DATOS ESPACIALES MEDIANTE UN SIG

Uno de los aspectos fundamentales de un sistema de localización, e íntimamente relacionado a la vez con el problema del conocimiento espacial, es cómo representar la información del entorno de un modo escalable, interoperable y compatible con el ser humano [108]. Los sistemas de información geográfica (SIG) son probablemente la única tecnología existente capaz de cumplir estos tres requisitos:

- Son capaces de representar la información a cualquier escala, desde los muebles de una habitación en un edificio, hasta información a nivel de naciones, como por ejemplo áreas de cultivo, bosques, etc.
- Su interoperabilidad con otros estándares es cada vez mayor. Gracias a recientes esfuerzos de compatibilidad por parte de la industria, hoy en día se pueden utilizar junto con sistemas de gestión de infraestructuras asistidos por computador (CAFM), sistemas de gestión integrada del espacio de trabajo (IWMS), modelado de información de construcción (BIM) o diseño asistido por computador (CAD), entre otros.

- Son completamente compatibles y fácilmente interpretables por el ser humano, porque en sus orígenes fueron diseñados con la intención de analizar y visualizar datos geoespaciales. Además la información se puede modelar simbólicamente y separar en capas independientes.

La información espacial es difícil de obtener, y la interoperabilidad permite a los SIG reutilizar modelos espaciales de lugares previamente modelados. Al integrar un sistema de localización con un SIG, se abre la posibilidad por ejemplo, de mantener localizado al robot en cualquiera de los sistemas con los que un SIG sea compatible, como por ejemplo en sistemas de gestión de infraestructuras. De este modo se podría incluir al robot como un recurso más y facilitar su gestión.

En nuestro caso aprovechamos la interoperabilidad de los SIG para obtener la información de modelado de un edificio via CAD, y utilizarla como mapa del entorno del robot. El modelo espacial del edificio queda así introducido en la base de datos espacial asociada al SIG, y el robot puede hacer consultas espaciales preguntando qué puertas son visibles desde su posición. Esta consulta, junto con la detección de puertas son dos procesos fundamentales para efectuar la *asociación de datos*, y que permiten al EKF corregir su posición, tal y como se indica en el esquema de la figura 5.1.1.

Otra opción sería utilizar un modelo BIM, como Borkowski et al. [18]

En el capítulo 5.4 se analiza en detalle cómo introducir un modelo CAD en un SIG y cómo realizar consultas a una base de datos espacial.

5.1.4 LOCALIZACIÓN

Al igual que sucede al aplicar cualquier filtro de Kalman, la aplicación del EKF para resolver el problema de la localización de un robot móvil consiste en un proceso cíclico de dos fases: *predicción* y *corrección*. En la figura 5.1.6 se muestra este ciclo de forma muy concisa y aplicado a nuestro problema de localización, aunque será explicado con detalle en el capítulo 5.5.

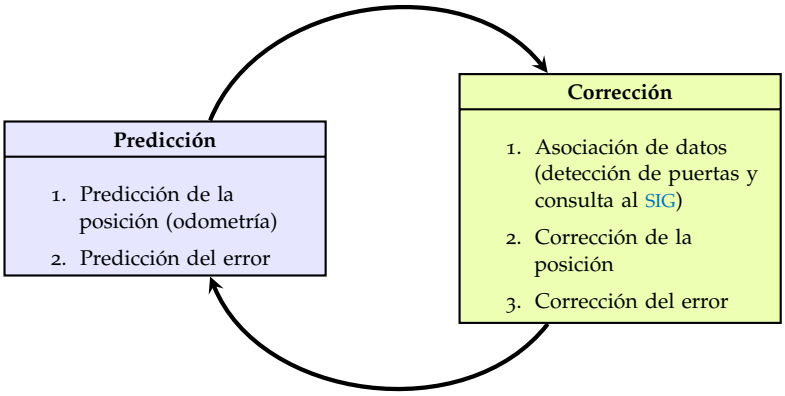


Figura 5.1.6: Funcionamiento cíclico del EKF aplicado a la localización de un robot móvil.

5.1.4.1 *Fase de predicción*

En esta fase, se aplica el modelo odométrico del sistema (capítulo 3.3) al desplazamiento medido por los *encoders* durante un incremento de

tiempo. Esto permite obtener una estimación de la posición del robot al transcurrir un instante de tiempo. Además de la posición, también se actualiza el error en la posición, puesto que los filtros de Kalman son probabilísticos y el estado se modela mediante variables aleatorias gaussianas.

El problema de esta fase es que la información odométrica es una medida propioceptiva, y esto implica que no depende de referencia externa y el error en la posición aumenta ilimitadamente a medida que transcurre el tiempo. Por lo tanto, resulta imprescindible corregir la estimación de la posición mediante medidas más fiables que hagan referencia a alguna característica externa, consiguiendo así acotar el error. De ello se encarga la fase de corrección del EKF.

5.1.4.2 *Fase de corrección*

La etapa de corrección es bastante más compleja que la de predicción. Esto se debe principalmente a que depende de uno del aspectos más críticos en lo que respecta al problema de la localización: el proceso de asociación de datos. Este proceso consiste básicamente en establecer una correspondencia entre las características del entorno observadas por el robot y las mismas características almacenadas en el mapa. La principal dificultad reside en la posibilidad de efectuar asociaciones erróneas, que en ocasiones pueden causar la divergencia del filtro y que el robot se pierda por completo. Existen diferentes técnicas de asociación de datos, y se ha optado por utilizar una búsqueda del vecino más próximo (*nearest neighbor)* basada en puertas de validación y distancias de Mahalanobis.

En nuestro caso la asociación de datos depende directamente de las puertas detectadas por el sensor virtual del robot (capítulos 5.2 y 5.3), así como del sistema de información geográfica en el que se almacena el mapa del entorno (capítulo 5.4). Las puertas observadas están en coordenadas robocéntricas, mientras que las puertas del mapa están en coordenadas globales. A través del modelo de observación (capítulo 3.4) se predice la posición de las puertas almacenadas en el SIG, es decir, se transforman a coordenadas robocéntricas. De este modo la asociación de datos se puede llevar a cabo, y se calcula una corrección de la posición. Además, al igual que en la fase de predicción, se corrige también el error en la estimación de la posición.

CAPÍTULO 5.2

EXTRACCIÓN DE CARACTERÍSTICAS CON LÁSER

En este capítulo se describe el método que proponemos para detectar las paredes de un pasillo utilizando un escáner láser. Para definir matemáticamente las paredes se necesita estimar la orientación relativa de la pared con respecto al robot, así como la distancia a la que se encuentra. Gracias al uso de histogramas, calculamos ambos parámetros de una forma rápida y eficiente. En el siguiente capítulo se describe cómo utilizar las paredes detectadas para calcular regiones de interés en las imágenes y acelerar el proceso de detección de puertas.

5.2.1 DEFINICIÓN Y EJEMPLO DE BARRIDO LÁSER

En cada instante de tiempo t, el escáner del robot obtiene un *barrido láser*, es decir, un conjunto de datos abarcando un semicírculo definido como:

$$\mathcal{S} = \{s_1, \ldots, s_n \mid s_i = (\rho_i, \varphi_i)\} \tag{5.2.1}$$

donde ρ_i y φ_i son las coordenadas polares del i-ésimo punto y el sensor es el origen del sistema de referencia. El láser, utilizando una resolución angular fija $\Delta\varphi = \varphi_i - \varphi_{i-1}$, obtiene los puntos secuencialmente y en sentido anti-horario, comenzando en $\varphi_{min} = \varphi_1$ y terminando en $\varphi_{max} = \varphi_n$. A lo largo de este trabajo se ha utilizado $n = 361$, $\Delta\varphi = 0.5°$, y una amplitud de barrido $\varphi_{max} - \varphi_{min} = 180°$.

En la figura 5.2.1 se muestra un ejemplo concreto de barrido láser. El robot está ubicado en el origen de coordenadas y su dirección de avance coincide con la dirección del eje Y.

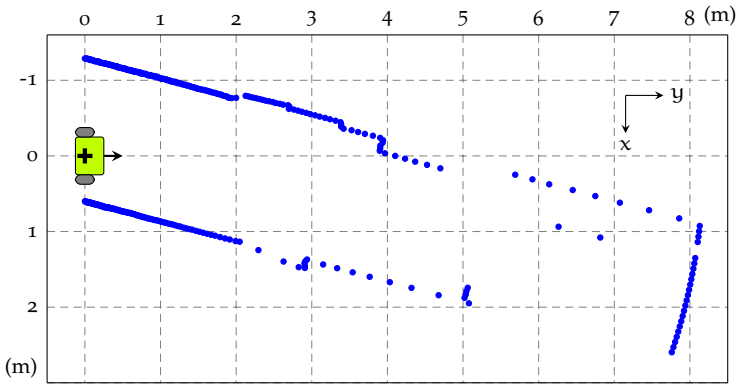


Figura 5.2.1: Ejemplo de barrido láser en un pasillo de la facultad.

5.2.2 CÁLCULO DE LA ORIENTACIÓN DE UN PASILLO

Para encontrar el ángulo α que el pasillo forma con la dirección de avance del robot, se ha optado por utilizar *histogramas angulares*. La idea de histograma angular se ha extraído del trabajo de *Weiß et al.* [112], pero hemos realizado una modificación para obtener unos mejores resultados.

5.2.2.1 *Histograma angular*

La técnica original consiste en transformar un barrido láser $\mathcal{S}$ a coordenadas cartesianas. Así, los puntos se pueden interpretar como un conjunto de vectores $\vec{\mathcal{V}}$ cuyo inicio es el origen de coordenadas:

$$\vec{\mathcal{V}} = \left\{ \vec{v_1}, \ldots, \vec{v_n} \;\middle|\; \vec{v_i} = \left(\rho_i \cdot \cos(\varphi_i), \rho_i \cdot \sin(\varphi_i)\right) \right\}$$

y donde las coordenadas polares del punto (ρ_i, φ_i) coinciden con el módulo y dirección de $\vec{v_i}$. La diferencia entre dos vectores consecutivos es también un vector y calculada sobre $\vec{\mathcal{V}}$ produce otro conjunto:

$$\vec{\mathcal{W}} = \left\{ \vec{w}_1, \ldots, \vec{w}_{n-1} \;\middle|\; \vec{w}_i = (\vec{v}_{i+1} - \vec{v}_i) \right\} \tag{5.2.2}$$

El histograma angular es un histograma en el que se acumulan las direcciones de los vectores del conjunto $\vec{\mathcal{W}}$. Cuando un barrido láser registra superficies planas, los vectores de $\vec{\mathcal{W}}$ son prácticamente colineales, y el histograma presenta máximos locales en la dirección principal de la mayoría de elementos $\vec{w}_i$. Por lo tanto, gracias a un histograma angular se puede averiguar la dirección principal de alineamiento de los puntos de un barrido láser.

Uno de los problemas de utilizar vectores consecutivos para calcular $\vec{\mathcal{W}}$ es que muy cerca del sensor la densidad de puntos es muy alta y la longitud de los vectores puede ser incluso $\|\vec{w}_i\| \leqslant 1\text{mm}$. Para vectores tan pequeños, el ruido en la medida del láser ($\sigma_l \approx 1 - 5\text{mm}$) es del orden de la longitud del vector, y esto introduce grandes errores en la dirección calculada. Sin embargo este error desaparece cuando σ_l es despreciable en comparación con $\|\vec{w}_i\|$. Como solución, *Weiß et al.* sugieren calcular los elementos de $\vec{\mathcal{W}}$ como la diferencia entre vectores no consecutivos:

$$\vec{\mathcal{W}} = \left\{ \vec{w}_1, \ldots, \vec{w}_{n-j} \;\middle|\; \vec{w}_i = (\vec{v}_{i+j} - \vec{v}_i) \right\} \tag{5.2.3}$$

donde j se debe decidir experimentalmente y es fijo para todos los $\vec{w}_i$. Al calcular la diferencia entre un vector y su j-ésimo sucesor, las direcciones de los vectores resultantes están menos influenciadas por ruido.

5.2.2.2 *Propuesta de histograma angular mejorado*

Después de realizar numerosas pruebas experimentales utilizando la ecuacion (5.2.3), en algunos casos los máximos no son totalmente claros y la dirección principal calculada no es la correcta. Esto se debe a que la separación entre vecinos no consecutivos, j, es un valor fijo y no garantiza que los vectores $\vec{w}_i$ tengan una longitud mínima.

La solución que proponemos consiste en modificar la ecuación (5.2.3) considerando una métrica basada en la distancia euclídea:

$$\vec{\mathcal{W}} = \left\{ \vec{w}_1, \ldots, \vec{w}_t \;\middle|\; \vec{w}_i = (\vec{v}_{i+j} - \vec{v}_i),\ \|\vec{w}_i\| \geqslant d_h \right\} \tag{5.2.4}$$

Al igual que en el caso anterior, cada vector $\vec{w}_i$ se calcula como la diferencia entre vectores no consecutivos. Sin embargo, en lugar de utilizar un valor fijo para j, nosotros proponemos que j varíe para cada $\vec{w}_i$, de modo que el vector $\vec{v}_{i+j}$ sea el que cumpla que $\|\vec{w}_i\| \geqslant d_h$. Como j es variable, el número de elementos del conjunto, t, es variable y no se puede conocer a priori, puesto que depende de la distancia umbral escogida y del barrido láser en concreto. El valor de la distancia umbral d_h se debe decidir experimentalmente, y en nuestro caso lo hemos fijado en $d_h = 0.30m$. En la figura 5.2.2 se muestra un ejemplo gráfico de lo explicado.

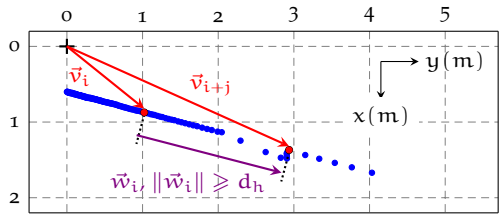


Figura 5.2.2: Ejemplo de cálculo de $\vec{w}_i$ en un fragmento del barrido láser de la figura 5.2.1

Al exigir que los elementos de $\vec{W}$ tengan una longitud mínima, se garantiza que el ruido en la medida del láser sea despreciable con respecto a la longitud de los vectores $\vec{w}_i$ y no afecte al cálculo de los ángulos.

5.2.2.3 *Algoritmo para calcular la dirección de un pasillo*

El algoritmo 5.2.1 resume lo explicado y muestra el procedimiento para calcular la dirección principal de un barrido láser. En primer lugar se calcula el conjunto de vectores $\vec{V}$ a partir del barrido láser original, y se construye un histograma h con un rango de -90^o a 90^o y barras de anchura b_w. A continuación, se calculan los vectores $\vec{w}_i$ usando la expresión (5.2.4) y se acumula su ángulo en la barra correspondiente del histograma. Finalmente, la dirección principal del pasillo se obtiene como la moda del histograma, cuyo cálculo se explica en el siguiente apartado.

Algoritmo 5.2.1 Cálculo de la dirección principal de un barrido

```
1:  procedure CalcularDirección(S)
2:      V ← polar_a_cartesianas(S)
3:      h ← histogram(−90, 90, b_w)
4:      for i ← 1, ··· , n − 1 do
5:          for j ← i + 1, ··· , n do
6:              w_i ← v_{i+j} − v_i
7:              if ‖w_i‖ ⩾ d_h then
8:                  Acumular(angulo(w_i), h)
9:              end if
10:         end for
11:     end for
12:     α ← moda(h)
13:     devolver α
14: end procedure
```

Un aspecto muy relevante a la hora de acumular los ángulos en el histograma es precisamente cómo calcular los ángulos. Este cálculo se puede realizar de diferentes maneras, y en nuestro caso se ha optado por hacerlo mediante la función *arctan*, con dominio $[-90, 90]^\circ$. Una particularidad de esta función es que no distingue entre direcciones diametralmente opuestas, y gracias a ello paredes enfrentadas aparecen en el histograma con el mismo ángulo (cosa que no ocurriría si se utilizara la función *atan2*). Así, en el caso de un pasillo, en el histograma existirá únicamente un máximo.

5.2.2.4 *Moda del histograma angular*

Una vez construido el histograma, se necesita calcular la moda, es decir, el valor que se repite con mayor frecuencia. La moda del histograma indica la orientación del pasillo con respecto al robot. Los datos del histograma angular son *datos agrupados* y por ello la moda del histograma no se debe calcular simplemente buscando el pico más alto. Cada una de las barras del histograma se denomina *intervalo de clase*, y el máximo absoluto del histograma se denomina *clase modal*. Cuando todos los intervalos de clase tienen el mismo tamaño, como sucede en el caso del histograma angular, la moda se calcula mediante la siguiente fórmula:

$$M = L + \left(\frac{f_1 - f_0}{2f_1 - f_0 - f_2} \right) b_w \tag{5.2.5}$$

donde L es el límite inferior de la clase modal; f_1 es la frecuencia absoluta de la clase modal; f_0 y f_2 son las frecuencias absolutas de los intervalos anterior y siguiente a la clase modal respectivamente, y b_w es la anchura del intervalo de clase.

En la figura 5.2.3 se muestra el histograma angular para el barrido láser de la figura 5.2.1, calculando los elementos de $\vec{W}$ como se indica en la expresión (5.2.4). Tras diferentes pruebas experimentales se ha

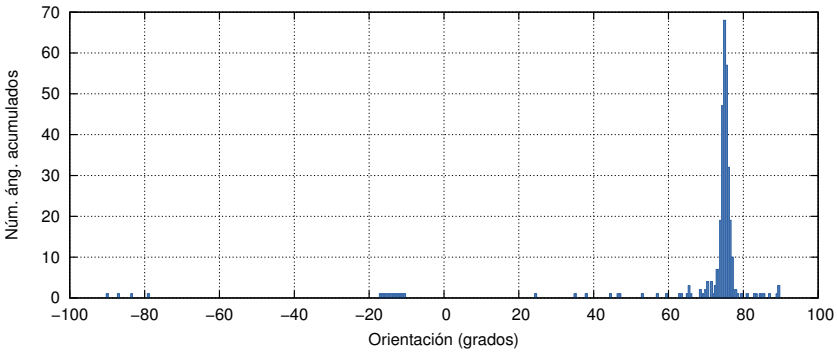


Figura 5.2.3: Histograma angular con barras de 0.5º de anchura correspondiente a la figura 5.2.1.

utilizado una anchura de intervalo de clase $b_w = 0.5$, lo que hace que el número de barras sea 360. La clase modal en este ejemplo es el intervalo $[74.75 - 75.25]$ grados. Si se tomase el centro de dicho intervalo como moda, el ángulo resultante sería $\alpha = 75^\circ$. Sin embargo, calculando la moda mediante la ecuación (5.2.5) resulta $\alpha = 75.09^\circ$. Gracias a esto, el ángulo estimado está en un intervalo continuo, y se elimina la discretización introducida al crear el histograma angular.

5.2.3 ALINEACIÓN DEL BARRIDO LÁSER

El siguiente paso para poder detectar y calcular la distancia a las paredes de un pasillo consiste en alinear el barrido láser con el eje Y. De este modo, el proceso de detección de las paredes se puede realizar de forma rápida y sencilla, tal y como se explicará en el siguiente apartado.

Sabiendo que la dirección principal calculada en el histograma angular es α, el conjunto de puntos $\mathcal{S}$ se rota de modo que quede alineado con el eje Y. Además, se transforma a coordenadas cartesianas para poder detectar posteriormente las paredes. Esta transformación da lugar a un nuevo conjunto de puntos $\mathcal{R}$ que se calcula como:

$$\begin{aligned} \mathcal{R} &= \{r_1, \ldots, r_n \mid r_i = (x_i, y_i)\} \\ \begin{bmatrix} x_i \\ y_i \end{bmatrix} &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix} \cdot \begin{bmatrix} \rho_i \cos(\varphi_i) \\ \rho_i \sin(\varphi_i) \end{bmatrix} \end{aligned} \tag{5.2.6}$$

Aquellos puntos que se encuentran a una distancia mayor de 8 metros se suprimen del conjunto $\mathcal{R}$, dado que estos puntos indican que el haz de luz emitido no ha regresado al sensor. En la figura 5.2.4, se muestra el resultado de rotar el barrido láser de la figura 5.2.1.

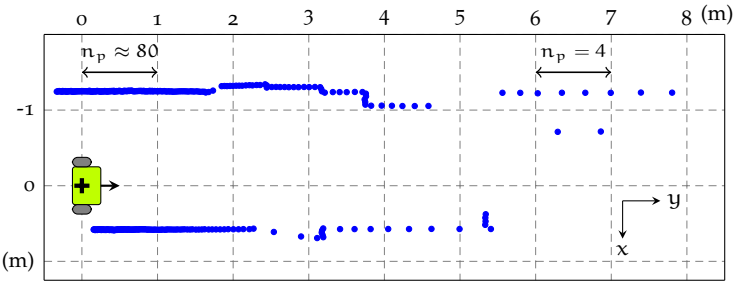


Figura 5.2.4: Barrido láser rotado y alineado con el eje Y,y eliminando los puntos a más de 8 metros.

5.2.4 DETECCIÓN DE PAREDES

Para detectar las paredes en un barrido láser no alineado con el eje Y habría que emplear técnicas costosas computacionalmente y que tuvieran tolerancia a *outliers*, tales como la transformada de Hough [36] o el método RANSAC [41], por ejemplo. Sin embargo, gracias a que el barrido láser $\mathcal{R}$ se ha alineado de forma paralela al eje Y, las dos paredes de un pasillo se pueden identificar fácilmente averiguando el valor de la coordenada x que se repite con mayor frecuencia en $\mathcal{R}$.

Un 'outlier' es una observación numéricamente distante del resto de los datos de un conjunto.

5.2.4.1 *Histograma-X*

Un método muy eficiente para calcular la coordenada x que se repite con mayor frecuencia es utilizar un *histograma*-X [112] en el que se acumulan las coordenadas x de los puntos de $\mathcal{R}$. Es una técnica robusta ante *outliers*, al igual que la transformada de Hough, o el método RANSAC. Mientras que estas dos técnicas son muy costosas computacionalmente, el histograma-X requiere muy pocos recursos computacionales.

Cuando las dos paredes de un pasillo están visibles en un barrido láser, el histograma-X tiene dos máximos indicando la distancia a

la pared izquierda y la pared derecha. Los máximos son claramente identificables porque la pared izquierda siempre tiene valores de x negativos, mientras que la pared derecha tiene valores de x positivos.

En un histograma-X, cada punto del barrido láser acumulado en el histograma incrementa la barra correspondiente en una unidad. Esto presenta el inconveniente de que los tramos de superficies más cercanos al láser tienen un mayor peso en el histograma, puesto que la densidad de puntos es mayor en las cercanías del láser. En la figura 5.2.4 se puede ver como para dos segmentos de la misma longitud ($1m$), el número de puntos (n_p) que contiene cada uno varía enormemente en función de la distancia al origen. En las cercanías del sensor hay 80 puntos en un metro, mientras que a $6m$ de distancia solamente hay 4. En consecuencia, los puntos más cercanos al sensor determinarían el resultado del histograma-X y podrían producirse errores en la detección de la pared cuando la superficie más próxima correspondiese a otro objeto distinto a la pared.

5.2.4.2 *Propuesta de histograma-X mejorado*

La solución que proponemos al problema del histograma-X consiste en obtener un histograma en el que se acumule la longitud de las superficies en lugar del número de puntos de dicha superficie. Mediante esta modificación, la clase modal del histograma indica cuál es la superficie de mayor longitud, y no importa si dicha superficie está cerca o lejos del sensor, porque se acumula la longitud y no simplemente el número de puntos.

La base de esta técnica es acumular en el histograma las distancias entre cada par de puntos consecutivos. Para no introducir errores en el histograma se necesita una etapa de preprocesamiento de los datos del láser en la que se detecten los puntos de ruptura (*breakpoints*). Un punto de ruptura se puede definir como una medida del láser asociada a una discontinuidad ocurrida durante el proceso de escaneado [17]. Existen diferentes procedimientos para calcularlos, y en este caso hemos utilizado un algoritmo de detección de puntos de ruptura diseñado en nuestro Grupo de Robótica y denominado *Distance-based Convolution Clustering* (DCC) (Clustering por Convolución de Distancias) [39].

Para explicar de manera más clara el procedimiento, se utilizará el barrido láser rotado de ejemplo. En la figura 5.2.5 se muestran etiquetados manualmente como $a - g$ los diferentes segmentos en que se podría dividir la pared izquierda. Los tramos a, d y f corresponden a la pared principal. Además, se muestra una línea que indica el orden en

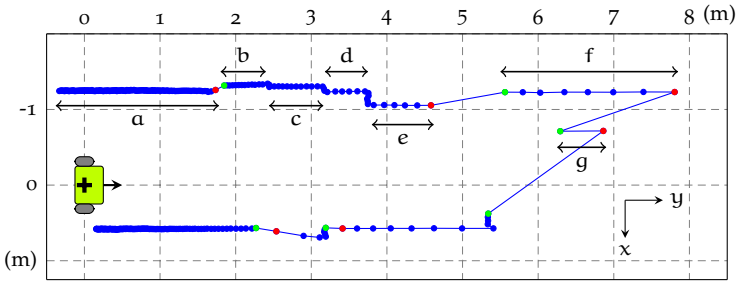


Figura 5.2.5: Etiquetado de superficies ($a - g$) para la pared izquierda. Los puntos de ruptura se indican mediante verde (inicio) y rojo (fin).

que el láser ha adquirido los puntos, empezando por la pared derecha (parte inferior izquierda de la figura) y en sentido antihorario. Los puntos verdes y rojos indican el comienzo y el fin, respectivamente, de un punto de ruptura en el proceso de escaneado. Por ejemplo, entre el final del segmento g y el comienzo del f existe un punto de ruptura que indica que la distancia entre ambos no debe ser acumulada en el histograma. Aquellos puntos que estén más distantes en la dirección del eje Y que en la dirección del eje X, tampoco se acumulan.

Después de acumular las distancias tal y como se ha descrito, el histograma-X para la pared izquierda se muestra en la figura 5.2.6, donde la anchura de cada barra se ha fijado en $b_w = 1$ cm. Se pueden

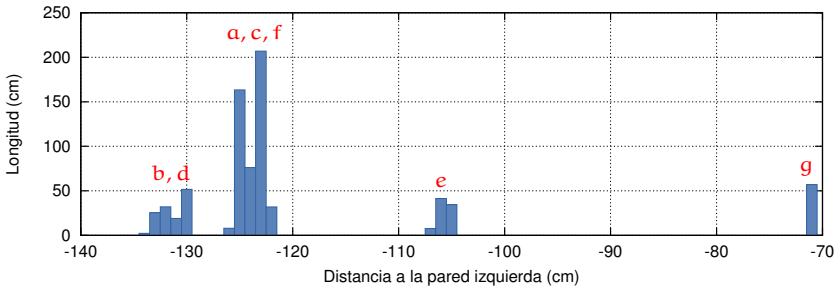


Figura 5.2.6: Histograma-X con barras de 1 cm de anchura para la pared izquierda de la figura 5.2.5. La moda se encuentra en −123.1 cm.

diferenciar claramente 4 grupos de barras, y para cada uno de ellos se han etiquetado las superficies correspondientes de la figura 5.2.5. El grupo principal corresponde a las superficies a, c, f, capturadas en un rango de 3cm (3 barras).

La moda del histograma-X se calcula del mismo modo que para el histograma angular, tal y como se explicó en el apartado 5.2.2.4, y en este caso está situada en −123.1 cm. El histograma-X para la pared derecha se construye de manera similar, obteniendo una moda de 57.3 cm, como se muestra en la figura 5.2.7.

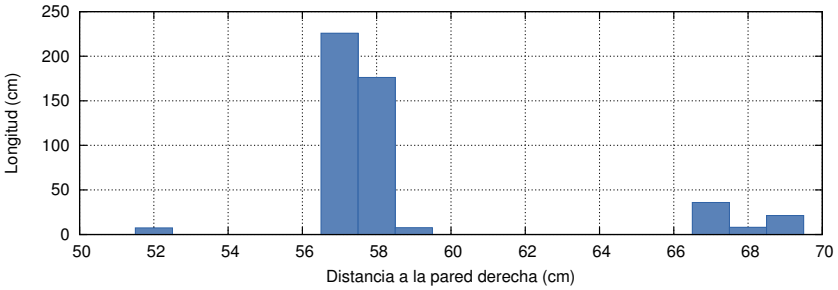


Figura 5.2.7: Histograma-X con barras de 1 cm de anchura para la pared derecha de la figura 5.2.5. La moda se encuentra en 57.3 cm.

5.2.4.3 *Algoritmo para la detección de una pared*

En un barrido láser rotado y alineado con el eje Y, la pared izquierda tiene coordenadas x negativas, mientras que la pared derecha tiene coordenadas x positivas. Por esta razón los algoritmos para detectar ambas paredes son ligeramente diferentes.

El algoritmo 5.2.2 resume cómo detectar y calcular la distancia a la pared izquierda. Si cada punto r_i de $\mathcal{R}$ tiene coordenadas (r_i^x, r_i^y), en

primer lugar se crea un subconjunto de puntos $\mathcal{N}$ eligiendo aquellos puntos de $\mathcal{R}$ con coordenadas r_i^x negativas. A continuación, se construye un histograma h cuyo rango es $[x_{min}, x_{max}]$ y con una anchura de barra b_w. Una vez hecho esto, se calcula el conjunto de puntos de ruptura, $\mathcal{B}$ mediante el algoritmo DCC comentado en el apartado anterior. Si el punto n_i es un punto de ruptura con respecto al punto n_{i-1}, entonces el elemento b_i de $\mathcal{B}$ toma el valor *verdadero*. En caso contrario valdrá *falso*.

Después de estos cálculos preliminares, se procede a acumular las distancias entre puntos consecutivos. Si la diferencia de coordenadas y entre dos puntos es mayor que la diferencia de coordenadas x, y no existe un punto de ruptura, entonces se acumula la distancia euclídea en el histograma. Por último, se calcula la moda con la expresión (5.2.5), que establece la distancia a la pared izquierda (δ_i).

Algoritmo 5.2.2 Detectar y calcular la distancia a la pared izquierda

```
1:  procedure PARED_IZQUIERDA(R)
2:      N ← puntos de R con r_i^x < 0          ▷ N ← {n_1, ··· , n_m}
3:      x_min ← mín(N^x)
4:      x_max ← máx(N^x)
5:      m ← num_puntos(N)
6:      h ← histograma(x_min, x_max, b_w)
7:      B ← dcc_breakpoints(N)                 ▷ B ← {b_1, ··· , b_m}
8:      for i ← 1, ··· , m − 1 do
9:          if (n_{i+1}^y − n_i^y) > (n_{i+1}^x − n_i^x) then
10:             if b_{i+1} es falso then
11:                 incrementar(h, dist_euclidea(n_i, n_{i+1}))
12:             end if
13:         end if
14:     end for
15:     return moda(h)                         ▷ Distancia a la pared izquierda
16: end procedure
```

La detección y el cálculo de la distancia a la pared derecha (δ_d) se realiza de manera similar, pero cambiando el conjunto $\mathcal{N}$ por un conjunto $\mathcal{P}$ que contenga únicamente los valores de $\mathcal{R}$ con x positivas. En la figura 5.2.8 se muestra el resultado de la detección de las paredes aplicado al barrido láser de ejemplo utilizado en este capítulo.

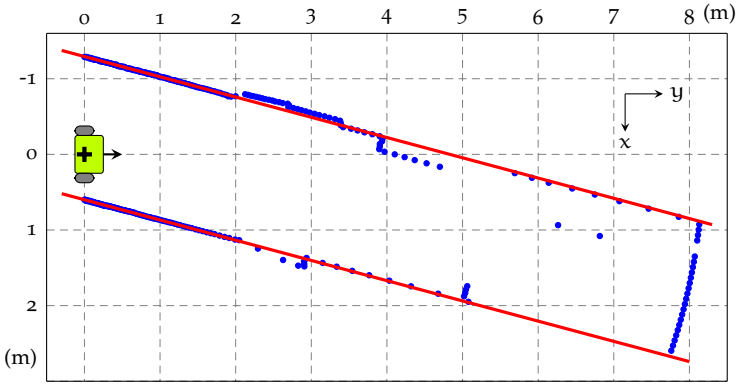


Figura 5.2.8: Paredes detectadas y superpuestas (en rojo) encima del barrido láser original.

CAPÍTULO 5.3

DETECCIÓN DE PUERTAS

En este capítulo se describe el método que proponemos para detectar puertas en las imágenes tomadas por la cámara a bordo del robot, así como para calcular la posición del centro de las puertas con respecto al robot. En primer lugar se describe la construcción de regiones de interés de la imagen a partir de las paredes detectadas por láser en el capítulo anterior. A continuación, se presenta la utilización de características pseudo-Haar para detectar líneas verticales correspondientes a los laterales de las puertas. Finalmente, se describe como detectar puertas de simple y doble marco a partir de las líneas verticales. Los métodos se han formalizado matemáticamente y están especificados mediante algoritmos.

5.3.1 CÁLCULO DE LAS REGIONES DE INTERÉS

En el capítulo anterior se han descrito los algoritmos necesarios para estimar la orientación del pasillo con respecto al robot (α), así como para detectar y calcular la distancia a las paredes derecha (δ_d) e izquierda (δ_i). Gracias al procedimiento de calibración descrito en el capítulo 4.2 y aplicado a nuestro robot, cualquier medida del láser o incluso cualquier punto del mundo real se puede proyectar en las imágenes obtenidas por la cámara. Por lo tanto es posible proyectar las paredes detectadas por el láser en la imagen.

En el sistema de referencia bidimensional del láser $\{X_L, Y_L\}$, las paredes izquierda y derecha son dos rectas que en forma normal se definen como:

$$r_i: \quad x_l cos(\alpha) + y_l sin(\alpha) = \delta_i \tag{5.3.1}$$
$$r_d: \quad x_l cos(\alpha) + y_l sin(\alpha) = \delta_d \tag{5.3.2}$$

Cada una de estas rectas corresponde a la vista en planta de una pared en el mundo real. Para poder proyectar cada una de estas rectas en la imagen, primero se debe definir cada pared como un plano en el sistema de referencia tridimensional del robot $\{X_R, Y_R, Z_R\}$:

$$\Pi_i: \quad x_r cos(\alpha) + y_r sin(\alpha) = \delta_i \tag{5.3.3}$$
$$\Pi_d: \quad x_r cos(\alpha) + y_r sin(\alpha) = \delta_d \tag{5.3.4}$$

Los planos Π_i y Π_d son verticales e infinitos, de modo que la coordenada z_r puede tomar cualquier valor, y para calcular las regiones de interés basta cortarlos con los planos:

$$\Pi_f: \quad z_r = 0 \tag{5.3.5}$$
$$\Pi_t: \quad z_r = 2.0 \tag{5.3.6}$$

donde Π_f es el plano del suelo y Π_t es un plano paralelo al suelo pero a la altura de la parte superior de la puerta (2 metros). El resultado de estas intersecciones son cuatro líneas en un espacio tridimensional que definen la parte inferior y superior de las RDI donde encontrar las puertas.

Estas líneas están definidas en el sistema de referencia del robot $\{X_R, Y_R, Z_R\}$, y el último paso para proyectarlas en la imagen consiste en tomar dos puntos específicos de una recta y utilizar las ecuaciones (4.2.14) y (4.2.15). Así se obtienen cuatro líneas en la imagen, como se muestra en la figura 5.3.1:

- λ_f y λ_t: definen la parte inferior y la parte superior de la pared izquierda.
- ξ_f y ξ_t: definen la parte inferior y superior de la pared derecha.

Figura 5.3.1: Regiones de interés en la imagen, junto con las cuatro líneas utilizadas para definirlas.

5.3.2 CARACTERÍSTICAS PSEUDO-HAAR

Las características pseudo-Haar son características visuales utilizadas para el reconocimiento de objetos, y tienen su origen en las *wavelets* bidimensionales. Su principal ventaja es que tienen un coste computacional extremadamente bajo si se calculan utilizando la técnica de la imagen integral (ver apéndice D). A continuación se ofrece una concisa explicación de la relación entre las características pseudo-Haar y las *wavelets*.

El término wavelet procede del inglés 'wave' (onda) y '-let' (sufijo diminutivo). Traducido al castellano resulta en 'ondícula', aunque el término inglés es utilizado con mayor frecuencia que su versión en castellano.

5.3.2.1 *Wavelets 1D*

Las *wavelets* son funciones matemáticas que deben cumplir ciertas restricciones y que se utilizan como base para la representación de otras funciones o conjuntos de datos (como imágenes, por ejemplo). Se basan, por tanto, en una idea similar a la transformada de *Fourier*, que permite expresar cualquier función como una suma de senos y cosenos. La principal diferencia entre estos dos métodos es que las *wavelets* están localizadas en el tiempo y la frecuencia, mientras que la transformada de *Fourier* sólo está localizada en la frecuencia.

Tal y como su nombre indica, el aspecto de las *wavelets* se asemeja al de una onda debido a que una de las propiedades que deben satisfacer

es que su integral sea igual a cero, lo que implica que los valores de la función mayores que cero se deben compensar con valores por debajo de cero. Además, el sufijo *-let* indica que la función crece y decrece en un periodo finito de tiempo relativamente pequeño, en contraste con una onda de mayores dimensiones, como los senos y cosenos utilizados en la transformada de *Fourier*, que no están acotados en el tiempo sino que oscilan de forma infinita.

De entre los diferentes tipos de *wavelets* existentes, la primera y más simple fue propuesta por *Alfréd Haar* en 1909 y se denomina *wavelet* de *Haar* en su honor. Consiste en un pequeño pulso rectangular positivo seguido de uno negativo, y que en su conjunto se puede escalar y trasladar en el tiempo con el fin de poder procesar los datos a diferentes escalas o resoluciones (ver fig. 5.3.2).

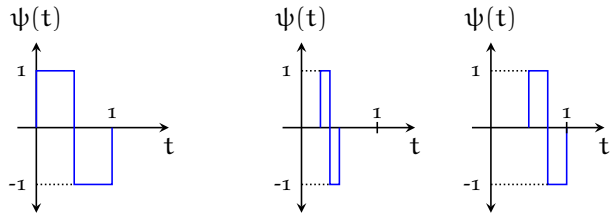


Figura 5.3.2: Ejemplos de escalado y traslación en una *wavelet* de *Haar*.

5.3.2.2 *Wavelets 2D*

La extensión natural de las *wavelets* de *Haar* de 1D a 2D se obtiene efectuando el producto tensorial de dos *wavelets* en 1D [86]. El resultado son las tres *wavelets* mostradas en la fig. 5.3.3. El primer tipo de *wavelet*

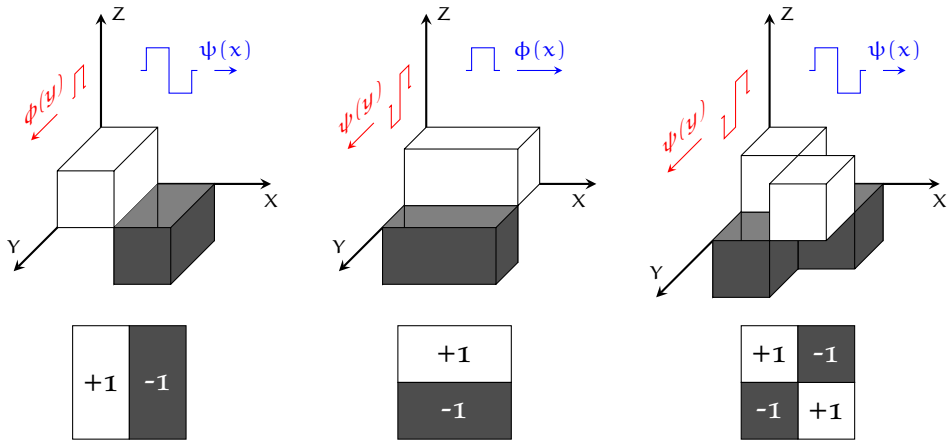


Figura 5.3.3: Los tres tipos de *wavelets* de *Haar* en 2D: vertical, horizontal y diagonal. Fila superior: vista 3D. Fila inferior: vista en planta, obteniendo las denominadas características pseudo-Haar.

es el producto tensorial de una wavelet por una función de escalado $\psi(x,y) = \psi(x) \otimes \phi(y)$, y codifica una diferencia de intensidad media a lo largo de un borde vertical. De manera similar el producto tensorial de una función de escalado por una wavelet, $\psi(x,y) = \phi(x) \otimes \psi(y)$ da lugar a una wavelet horizontal. Finalmente, el producto tensorial de dos wavelets $\psi(x,y) = \psi(x) \otimes \psi(y)$ da lugar a una wavelet diagonal.

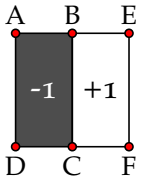
Al igual que las wavelets unidimensionales, las wavelets 2D se pueden escalar y trasladar.

5.3.2.3 *Características pseudo-Haar*

Basándose en la idea de *wavelets* bidimensionales y en el trabajo de *Papageorgiou* y *Poggio* [86], *Viola* y *Jones* [109] desarrollaron las 'características pseudo-Haar' *(Haar-like features)*. Reciben este nombre porque son visualmente similares a las *wavelets* 2D cuando se observan utilizando una vista en planta, como se muestra en la fila inferior de la figura 5.3.3.

Además de los 3 tipos mostrados en la figura 5.3.3, se pueden definir muchas otras características, como por ejemplo las propuestas por *Lienhart* [63]. No obstante, en nuestro detector de puertas únicamente necesitamos características pseudo-Haar verticales, que consisten en una ventana dividida verticalmente en dos regiones rectangulares adyacentes y del mismo tamaño. Esta ventana se debe colocar en una parte específica de la imagen, y a continuación se suman las intensidades de los píxeles de cada región y se calcula su diferencia. Este tipo de característica sirve para indicar dónde existe un borde entre una región clara y otra oscura, de modo que lo aprovecharemos para detectar los laterales de las puertas, como se explica en el siguiente apartado.

La principal ventaja de las características pseudo-Haar frente a otras alternativas es su velocidad de cálculo, que se debe al uso de la imagen integral. La imagen integral (ver apéndice D) se puede definir como una tabla de búsqueda en forma matricial con el mismo tamaño que la imagen original, donde cada elemento de la tabla (i, j) contiene la suma de todos los píxeles ubicados por encima y a la izquierda del pixel (i, j). Esto permite calcular la suma de los píxeles de cualquier área en tiempo constante, independientemente del tamaño, utilizando únicamente 4 referencias a la tabla de búsqueda que contiene los valores de la imagen integral. Utilizando esta propiedad, el valor correspondiente a una característica pseudo-Haar vertical se puede calcular mediante seis referencias al array de la imagen integral, tal y como se ilustra en la figura 5.3.4.



$$I \leftarrow \text{Imagen Integral}$$

$$V \leftarrow 2 \cdot I(B) + I(D) + I(F) - [I(A) + I(E) + 2 \cdot I(C)]$$

Figura 5.3.4: Cálculo del valor (V) de una característica pseudo-Haar vertical a partir de la imagen integral (I).

5.3.3 DETECCIÓN DE LOS LATERALES DE LAS PUERTAS

Una vez definidas las RDI, el siguiente paso de nuestro detector de puertas consiste en extraer líneas verticales en la imagen. Esta idea se basa en que en la mayoría de los casos, los laterales y la parte superior de las puertas son visualmente distinguibles de la pared en la que se encuentran. En una imagen, la parte superior de una puerta puede tener cualquier orientación, mientras que los laterales de las puertas

dan lugar a líneas verticales en la imagen. Es importante advertir que esto sólo ocurre cuando el eje óptico de la cámara está colocado horizontalmente; de lo contrario las líneas de los laterales de la puerta no serían verticales.

5.3.3.1 *Detección de líneas convencional*

Las líneas verticales han sido empleadas en multitud de trabajos dedicados a resolver el problema de localización de un robot móvil (ver apartado 1.2.4), si bien los autores han confiado en detectores de líneas *genéricos*. Algunas de las etapas más habitualmente empleadas en la detección de líneas son las siguientes:

Con detector de líneas 'genérico' nos referimos a un método que permita detectar líneas en cualquier orientación.

1. Suavizado de la imagen: típicamente se utilizan filtros de convolución gaussianos.
2. Detección de bordes: normalmente se utilizan filtros de convolución basados en el gradiente, como el operador de Sobel, Prewitt (ver apéndice C), u otros más sofisticados como el detector de bordes de Canny. Cada píxel que tiene las características de borde se denomina *edgel (edge element)*.
3. Etiquetado de componentes conectados (*connected components labeling*): determina la conexión entre los *edgels* obtenidos en la etapa anterior. Cada conjunto de *edgels* conectado y aislado de los demás se etiqueta con un identificador único.
4. Extracción de líneas: para cada conjunto de componentes conectados se comprueba si se puede ajustar a una línea, utilizando la transformada de Hough, el método de mínimos cuadrados u otras opciones.

No todos los detectores de líneas utilizan todos estos pasos, pero sí son bastante representativos. La conexión de *edgels* es un paso bastante costoso, puesto que requiere comprobar los *edgels* contiguos a uno dado. Incluso la aplicación de filtros de convolución para suavizar la imagen o detectar bordes se convierte en un proceso costoso a no ser que se trabaje con imágenes de muy baja resolución.

5.3.3.2 *Detección de líneas mediante características pseudo-Haar*

Para detectar los laterales de las puertas, no es imprescindible utilizar un detector de líneas genérico. De hecho, sabiendo que las líneas a extraer son perfectamente verticales, se ha desarrollado un sistema detección de líneas con un tiempo de computación extremadamente rápido y mucho más simple que cualquier otro existente. Todo ello sin utilizar filtros de convolución o conexión de componentes. Básicamente, la idea que proponemos consiste en utilizar características pseudo-Haar verticales a modo de ventanas deslizantes a lo largo de las RDI definidas en la imagen, como se muestra en la figura 5.3.5a.

Cada *ventana de Haar* está representada en la imagen como un área rectangular dividida verticalmente en dos mitades: una mitad negra que indica que los píxeles se deben restar y una parte blanca que indica que se deben sumar. El resultado de sumar y restar los píxeles de cada mitad es un valor distinto de cero cuando el centro de la ventana

De aquí en adelante utilizaremos el término 'ventana de Haar' para referirnos a una 'característica pseudo-Haar vertical', para simplificar la terminología.

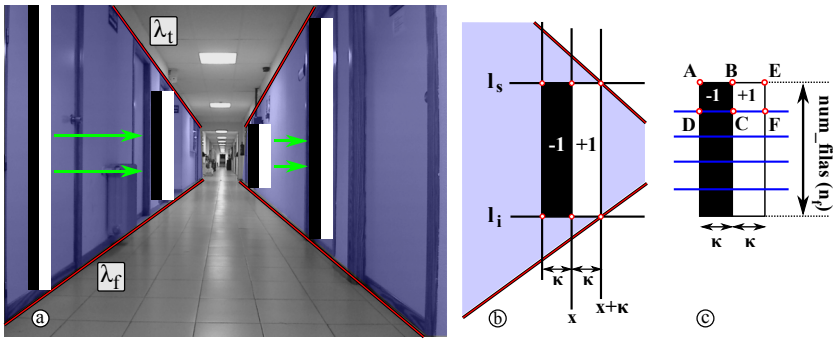


Figura 5.3.5: Detección de líneas utilizando características pseudo-Haar verticales en las regiones de interés previamente definidas.

coincide sobre una línea vertical, o prácticamente cero si está sobre un área más o menos uniforme, como por ejemplo una pared.

El algoritmo 5.3.1 detalla el pseudocódigo para calcular el valor de una ventana de Haar de anchura 2κ, centrada en el pixel x, siendo Ω el array que contiene la imagen integral. El algoritmo realiza el cálculo suponiendo que la ventana está ubicada en la RDI correspondiente a la pared izquierda, definida por las líneas λ_f y λ_t (ver fig. 5.3.5b). La altura de cada ventana se adapta de modo que su parte derecha se corte con λ_f y λ_t. Es necesario utilizar la parte derecha de la ventana, ya que en otro caso se capturarían píxeles fuera de la RDI. Por tanto, para una ventana centrada en el pixel x, la línea vertical $x + \kappa$ se corta con las líneas λ_f y λ_t, obteniendo las líneas horizontales l_s y l_i. En la pared derecha la técnica es similar, pero utilizando la parte izquierda de la ventana $(x - \kappa)$.

Para hacer el algoritmo más robusto, cada ventana de Haar se subdivide en n_f filas (fig. 5.3.5c) en lugar de calcular la ventana total directamente. Para cada fila, se calcula una *subventana de Haar normalizada*: la diferencia entre las áreas BEFC y ABCD se normaliza dividiendo por la mitad del área en píxeles de la subventana. Si el valor normalizado es mayor que un valor umbral T_n, entonces significa que se ha detectado una línea vertical en esa subventana, y se incrementa el contador de sublíneas verticales n_s. Finalmente, si el porcentaje de sublíneas (n_s/n_f) es mayor que un valor umbral T_p, se devuelve la suma de todas las subventanas de Haar normalizadas. En caso contrario se devuelve cero. Al utilizar subventanas, el algoritmo es más robusto porque eligiendo n_f y T_p adecuadamente, se pueden detectar puertas ocultas parcialmente por algún otro objeto. Los valores concretos seleccionados experimentalmente para las constantes mencionadas se recogen en la tabla 5.3.1.

Par.	Valor	Definición
κ	3	Mitad de la anchura (en px) de una ventana de Haar.
n_f	6	Número de subventanas verticales para cada ventana de Haar.
T_n	10	Valor mínimo (en px) que debe tener una subventana para considerar que ha detectado una línea.
T_p	5/6	Porcentaje de subventanas que deben de detectar una línea vertical.

Tabla 5.3.1: Valores de las constantes utilizadas para calcular el valor de una ventana de Haar.

Algoritmo 5.3.1 Detección de líneas verticales

```
1:  procedure VENTANA_DE_HAAR_ADAPTATIVA(x, λ_f, λ_t, Ω)
2:      (l_s, l_i) ← calcular_lineas(x, λ_f, λ_t)
3:      sum ← 0
4:      n_s ← 0                          ▷ Contador de sublíneas verticales
5:      Δr = (l_s − l_i)/n_f
6:      for i ← 0 to n_f − 1 do
7:          A ← (x, l_s + i ∗ Δr)                          ▷ A = (A_x, A_y)
8:          D ← (x, A_y + Δr)
9:          temp ← Subventana_Normalizada(A, D, Ω)
10:         sum ← sum + temp
11:         if temp > T_n then
12:             n_s ← n_s + 1
13:         end if
14:     end for
15:     sum ← sum/n_f                                      ▷ Normalización
16:     if (n_s/n_f) < T_p then
17:         sum ← 0
18:     end if
19:     return sum
20: end procedure
21: procedure SUBVENTANA_NORMALIZADA(A, D, Ω)
22:     B ← (A_x + κ, A_y)
23:     C ← (D_x + κ, D_y)
24:     E ← (B_x + κ, B_y)
25:     F ← (C_x + κ, C_y)
26:     sum ← 2 · Ω(B) + Ω(D) + Ω(F) − [Ω(A) + Ω(E) + 2 · Ω(C)]
27:     return sum/area(ABCD)
28: end procedure
```

Si se calcula una ventana deslizante de izquierda a derecha en la imagen de ejemplo utilizando el algoritmo 5.3.1, entonces para cada píxel a lo ancho de la imagen se obtiene un valor que puede ser cero o distinto de cero, como se representa en la gráfica de la figura 5.3.6. Un valor distinto de cero indica que en ese píxel se ha detectado una línea vertical, y el valor es mayor cuanto mayor sea el contraste de la línea detectada. La normalización del valor calculado para cada ventana de Haar es un paso fundamental porque el alto de cada ventana es adaptativo. Sin normalizar el cálculo, el resultado del filtrado por valor umbral (T_i, línea 11) no resultaría efectivo.

Si se superpone esta gráfica sobre la imagen a partir de la que se calculó (ver figura 5.3.7), se puede ver que los picos coinciden perfectamente con las líneas verticales en la imagen. Los picos negativos se corresponden a cambios de intensidad de claro a oscuro, mientras que los picos positivos indican un cambio de intensidad de oscuro a claro. Esta circunstancia se puede aprovechar para buscar dónde empieza y dónde acaba una puerta, o el marco de una puerta. Por ejemplo, en el tipo de puertas que aparecen en la imagen, cada marco está definido por un par de picos negativo-positivo.

Suponiendo que se han obtenido t valores distintos de cero, se construye un conjunto de líneas verticales:

$$\mathcal{L}_1 = \{l_1, \ldots, l_t \mid l_i \equiv (x_i^I,\ I|F)\} \tag{5.3.7}$$

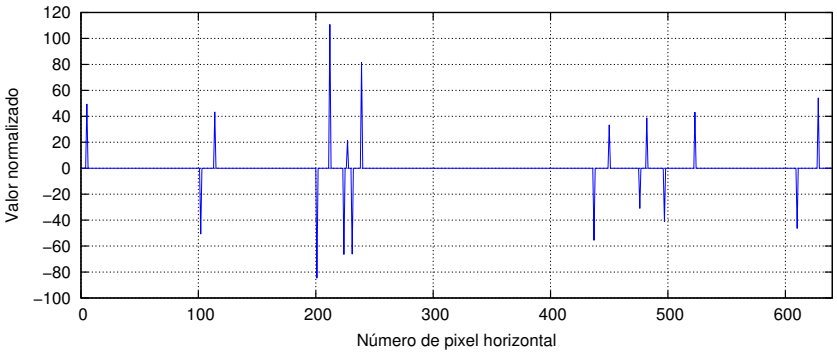


Figura 5.3.6: Valores resultantes de aplicar el algoritmo de ventanas deslizantes para detectar las líneas verticales dentro de las regiones de interés.

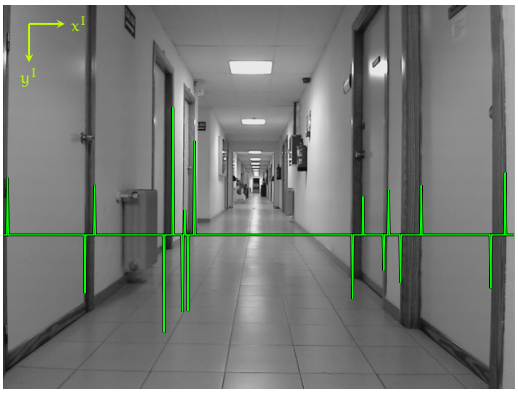


Figura 5.3.7: Superposición de los valores de la figura 5.3.6 sobre la imagen de ejemplo. Los picos coinciden con las líneas verticales en la imagen.

donde x_i^I es una constante en el sistema de coordenadas de la imagen $\{X_I, Y_I\}$, y que sirve para definir la línea vertical $x = x_i^I$. Además se almacena el valor I o F para indicar si se trata de una línea de *inicio* o de *fin*. Consideramos una línea de *inicio* la que representa un cambio de claro a oscuro (pico negativo) y una línea de *fin* la que representa un cambio de oscuro a claro (pico positivo). Esta decisión se ha tomado teniendo en cuenta que las paredes en general son de un color más claro que las puertas.

5.3.4 DETECCIÓN DE PUERTAS

Cada una de las líneas verticales del conjunto $\mathcal{L}_1$ (5.3.7) se pueden considerar como posibles laterales de una puerta. Es decir, algunas de las líneas detectadas realmente corresponden a una puerta, mientras que otras líneas pueden ser falsos positivos y corresponder a una estantería, un armario u otro objeto que se observe como una línea vertical en la imagen y tenga la altura definida en la RDI.

5.3.4.1 *Cálculo de la posición de las líneas verticales*

El objetivo que nos planteamos es averiguar las coordenadas (x_i^L, y_i^L) en el sistema de referencia del láser para cada línea vertical l_i de $\mathcal{L}_1$ (5.3.7). La solución que proponemos es buscar en la imagen los puntos láser proyectados a la izquierda y a la derecha de la línea, e interpolar

su posición entre los puntos láser reales. En la figura 5.3.8 se muestra el conjunto de líneas verticales detectadas junto con los puntos del barrido láser correspondiente proyectados en la imagen. Las líneas

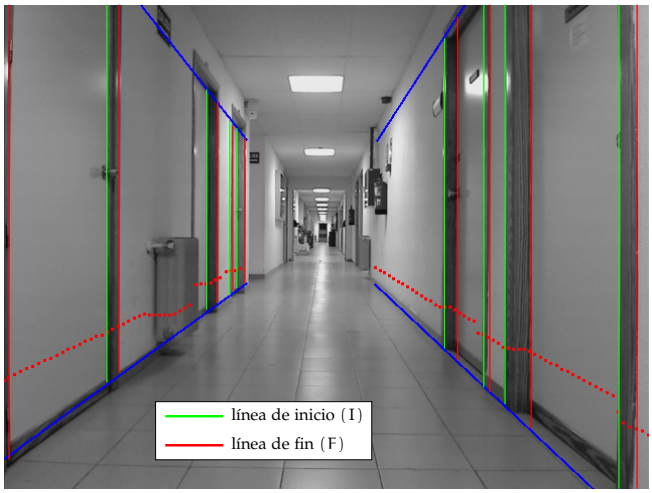


Figura 5.3.8: Líneas verticales a partir de los valores de la figura 5.3.6. Las líneas verdes corresponden a los picos negativos, mientras que las rojas corresponden a los picos positivos.

verdes corresponden a valores negativos al calcular la ventana de Haar (cambio de claro a oscuro, marcadas como *I*), mientras que las líneas rojas indican valores positivos (cambios de oscuro a claro, marcadas como *F*).

En primer lugar, hay que utilizar el proceso de calibración descrito en el capítulo 4.2 para proyectar el láser en la imagen. Así, un barrido láser formado por un conjunto de puntos:

$$\mathcal{S} = \{s_1, \ldots, s_n \mid s_j = (x_j^L, y_j^L)\} \tag{5.3.8}$$

donde (x_j^L, y_j^L) son las coordenadas cartesianas en el sistema de referencia del láser, da lugar a un conjunto de puntos proyectados en la imagen:

$$\mathcal{P} = \{p_1, \ldots, p_n \mid p_j = (x_j^I, y_j^I)\} \tag{5.3.9}$$

donde (x_j^I, y_j^I) son las coordenadas en píxeles de cada punto proyectado en la imagen. De este modo, se establece una equivalencia entre cada punto s_j del láser y su proyección en la imagen p_j:

$$f : \mathbb{R}^2 \to \mathbb{R}^2 \tag{5.3.10}$$

$$s_j \to p_j \tag{5.3.11}$$

donde la función de transformación $f(s_j) = p_j$ se calcula mediante ecuaciones (4.2.14) y (4.2.15).

Una vez establecida esta proyección, para cada línea vertical l_i se deben buscar dos puntos láser del conjunto de puntos proyectados ($\mathcal{P}$):

1. El punto p_a tal que $x_a^I \leqslant x_i^I$ (el más cercano a l_i por la izquierda).
2. El punto p_b tal que $x_b^I \geqslant x_i^I$ (el más cercano a l_i por la derecha).

Tras encontrar estos dos puntos, se obtienen sus correspondientes del barrido láser original: s_a y s_b. Esto significa que la línea l_i en la imagen

está ubicada entre los puntos s_a y s_b del barrido láser. Finalmente, calculando la posición relativa de l_i entre p_a y p_b se puede interpolar su posición entre s_a y s_b, obteniendo su posición final (x_i^L, y_i^L) en el sistema de referencia del láser. De este modo, se puede redefinir el conjunto de líneas $\mathcal{L}_1$ como:

$$\mathcal{L}_2 = \{l_1, \ldots, l_t \mid l_i \equiv (x_i^I,\ I|F,\ x_i^L,\ y_i^L)\} \tag{5.3.12}$$

donde (x_i^L, y_i^L) son las coordenadas de la línea l_i en el sistema de referencia del láser.

5.3.4.2 *Detección de puertas con doble marco*

El tipo de puertas que aparecen en la figura 5.3.8 tiene dos marcos laterales formado cada uno de ellos por una línea de inicio (I) seguida de una línea de fin (F). Cada marco lateral se puede identificar mediante el autómata finito determinista de la figura 5.3.9, utilizando el conjunto de líneas $\mathcal{L}_2$ como entrada y la propiedad de cada línea que indica si es una línea (I) o (F). Un marco comienza cuando se detecta la primera línea de inicio, y se acepta cuando se detecta una línea de fin, con la posibilidad de que entre medias se encuentren más líneas de inicio. Esto da lugar a un conjunto de m marcos:

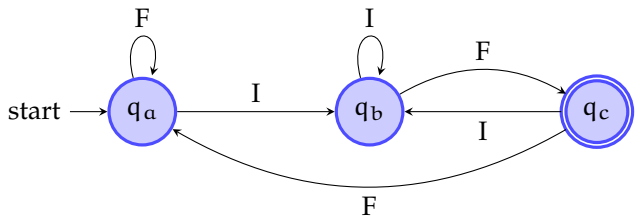


Figura 5.3.9: Automata finito determinista para la detección de marcos laterales de una puerta (I = línea de inicio, F = línea de fin).

$$\mathcal{M} = \{m_1, \ldots, m_m \mid m_i \equiv (l_s, l_e)\} \tag{5.3.13}$$

donde cada marco m_i está formado por una línea de inicio l_s y una línea de fin l_e ambas pertenecientes al conjunto de líneas $\mathcal{L}_2$.

Como el conjunto de líneas $\mathcal{L}_2$ contiene la ubicación de cada línea en el sistema de referencia del láser, el último paso consiste en comprobar si la distancia euclídea entre el centro de dos marcos m_i y m_j está dentro de unos límites. En dicho caso, se considera que se ha detectado una puerta correctamente y se almacenan como centro de la puerta las coordenadas del punto medio entre los marcos m_i y m_j. En la implementación de nuestro sistema aceptamos marcos entre $T_{min} = 0.8$ y $T_{max} = 1.10$ metros. El resultado de aplicar esta técnica es un conjunto de d puertas:

$$\mathcal{D} = \{d_1, \ldots, d_d \mid d_i \equiv (x_i^L, y_i^L)\} \tag{5.3.14}$$

donde (x_i^L, y_i^L) son las coordenadas del centro de la puerta en el sistema de referencia del láser. El algoritmo 5.3.2 especifica los detalles concretos del método.

En la figura 5.3.10 se muestra el resultado de la detección de puertas. En la pared derecha se han efectuado dos detecciones acertadas. En la pared izquierda, sin embargo, la primera puerta no se ha podido

Algoritmo 5.3.2 Detección de puertas con marco doble

```
1:  procedure PUERTAS_MARCO_DOBLE(M, m)
2:      D ← {∅}
3:      for i ← 1 to m − 1 do
4:          p_1^L ← punto_medio(m_i.l_s, m_i.l_e)
5:          for j ← i + 1 to m do
6:              p_2^L ← punto_medio(m_j.l_s, m_j.l_e)
7:              h ← dist_euclídea(p_1^L, p_2^L)
8:              if T_min ⩽ h ⩽ T_max then
9:                  añadir[D, punto_medio(p_1^L, p_2^L)]
10:                 i ← j + 1
11:                 exit for loop
12:             end if
13:             if h > T_max then
14:                 exit for loop
15:             end if
16:         end for
17:     end for
18:     return D
19: end procedure
```

detectar al no observarse completamente el marco lateral izquierdo, y la segunda puerta está parcialmente oculta por el radiador. En la tercera puerta se han detectado correctamente los dos marcos laterales, pero la medida de la anchura por láser no está dentro de los márgenes establecidos. Esto se debe a que la resolución del láser no es ofrece suficiente precisión a distancias lejanas y cuando la puerta es observada con un ángulo de visión muy cerrado.

Figura 5.3.10: Resultado de la detección de puertas: en la pared derecha se han efectuado dos detecciones, mientras que en la pared izquierda no se ha podido detectar ninguna puerta.

5.3.4.3 *Detección de puertas con marco simple*

Además de las puertas con marco doble, en las que se distingue claramente un marco a cada lado de la puerta, se puede considerar otro tipo de puertas, como la que se muestra en la parte izquierda de la figura 5.3.11. En este caso, el color de la puerta y el marco de la puerta coinciden, y es bastante probable que al aplicar el algoritmo 5.3.1 a este tipo

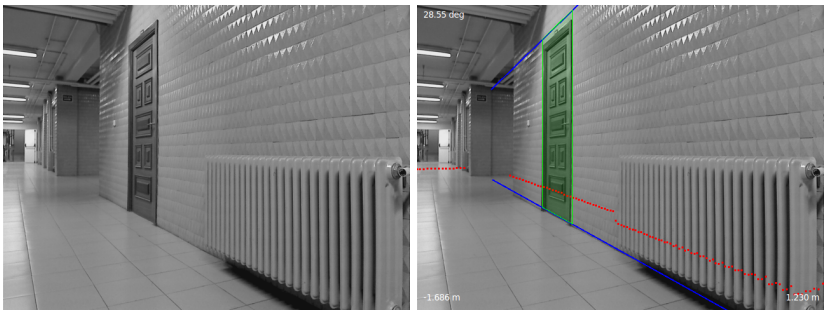


Figura 5.3.11: Izquierda: puerta con marco simple. Derecha: resultado de la detección.

de puertas se detecte una única línea a cada lado de la puerta. A estas puertas las denominamos puertas con marco simple y proponemos un procedimiento de detección ligeramente diferente al descrito para las puertas con marco doble.

El método se describe en el algoritmo 5.3.3, y toma como entrada el conjunto de líneas $\mathcal{L}_2$ (5.3.12) así como el número de líneas en el conjunto, t. Básicamente, el algoritmo busca una línea de inicio (tipo I) que corresponde al comienzo de la puerta, y a continuación una línea de fin (tipo F), que corresponde al final de la puerta. Si la distancia euclídea entre ambas líneas está dentro de los límites establecidos, se calcula el punto medio en el sistema de referencia del láser, y el resultado se añade al conjunto final de puertas $\mathcal{D}$, al igual que en el caso de los marcos dobles. El resultado de la detección se muestra en la parte derecha de la figura 5.3.11.

Algoritmo 5.3.3 Detección de puertas con marco simple

```
1:  procedure PUERTAS_MARCO_SIMPLE(L_2, t)
2:      D ← {∅}
3:      for i ← 1 to t − 1 do
4:          if l_i es de tipo I then
5:              for j ← i + 1 to t do
6:                  if l_j es de tipo F then
7:                      h ← dist_euclídea(l_i, l_j)
8:                      if T_min ⩽ h ⩽ T_max then
9:                          añadir[D, punto_medio(l_i, l_j)]
10:                         i ← j + 1
11:                         exit for loop
12:                     end if
13:                     if h > T_max then
14:                         exit for loop
15:                     end if
16:                 end if
17:             end for
18:         end if
19:     end for
20:     return D
21: end procedure
```

CAPÍTULO 5.4

CONSULTAS ESPACIALES

Gracias al EKF, el robot siempre conoce una estimación de su posición. En cada posición, se necesita resolver el problema de la asociación de datos (ver apartado 5.5.4), que permite corregir la posición del robot al establecer una correlación entre las puertas que el robot observa desde su posición y las puertas que en realidad son observables desde la posición estimada. Para obtener las puertas que el robot debería observar desde una posición concreta, se consulta a la base de datos espacial el centroide de los objetos de tipo puerta que estén dentro de un cierto radio de distancia a partir de la posición estimada del robot.

Para poder efectuar las consultas espaciales, se han introducido los planos del entorno en una base de datos espacial PostGIS, como se explica en el primer apartado. A continuación se describe el formato de los datos que devuelven las consultas espaciales, así como el esquema conceptual de la base de datos, que es necesario comprender para construir las consultas. Finalmente se describen las consultas utilizadas para predecir las puertas observables desde la posición estimada del robot.

5.4.1 CONVERSIÓN DE UN PLANO A DATOS ESPACIALES

Un plano es un diagrama, normalmente a escala, que muestra una vista en planta de las relaciones entre los distintos espacios que conforman un edificio a un mismo nivel. Dependiendo del tipo de formato de mapa empleado (raster o vectorial), el contenido puede ser más o menos estructurado, haciendo la información más o menos accesible.

El estándar *de facto* para crear y almacenar planos de interiores es el formato *AutoCAD Drawing* (DWG) de Autodesk, Inc. Es un formato propietario y cerrado utilizado por varios programas CAD. Está diseñado para capturar y representar información geométrica para tareas técnicas como la arquitectura, producción industrial y prototipado. Mientras que Autodesk nunca ha hecho públicas las especificaciones del formato DWG, sí que lo ha hecho para el formato *AutoCAD Drawing Interchange File* (DXF), con el fin de favorecer la interoperabilidad entre AutoCAD y otras aplicaciones CAD que manejen formatos diferentes. Así, entre estos dos formatos, DXF es el preferido por la mayoría de programas debido a que no se necesita una licencia y las especificaciones del formato son publicadas y actualizadas regularmente por Autodesk.

5.4.1.1 *Servicio Transfronterizo de Información Geográfica*

Todas las plantas de los distintos edificios de la Universidad de Salamanca tienen asociado un plano digital en formato DXF, mantenido por el *Servicio Transfronterizo de Información Geográfica* (STIG) de la Univer-

sidad de Salamanca, que regularmente revisa y actualiza los planos cada vez que se lleva a cabo alguna modificación de los espacios de la universidad.

En la figura 5.4.1 se muestra un ejemplo de plano digital almacenado por el STIG correspondiente a una de las plantas de la Facultad de Farmacia. El espacio está modelado digitalmente usando una represen-

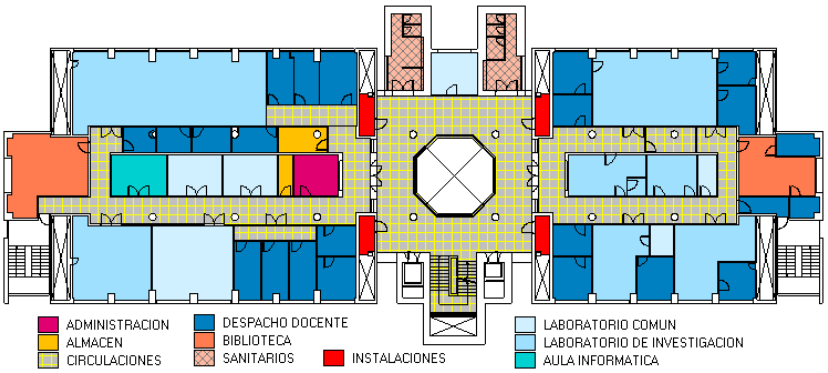


Figura 5.4.1: Ejemplo de plano digital de una de las plantas de la Facultad de Farmacia de la Universidad de Salamanca. Los diferentes espacios están categorizados por capas.

tación vectorial que contiene información semántica por capas, tal y como se indica en la leyenda. Como se puede observar, no solamente se almacena la información geométrica del entorno, sino que se conoce el uso que se espera de cada espacio: pasillos, laboratorios de investigación, aulas de informática, despachos de profesores, etc. Además también dispone del nombre de cada espacio, su área, y otros elementos como puertas, ventanas, tabiques, extintores de incendios, mobiliario, y otros; todo distribuido por capas.

5.4.1.2 *Importación de datos a PostGIS*

Para introducir los datos de un archivo DXF en una base de datos espacial PostGIS, en primer lugar se necesita convertir el fichero DXF a formato *shapefile*. Para efectuar esta conversión, se ha utilizado el programa Autocad Map 3D. El proceso de conversión se describe brevemente a continuación dado que no es demasiado intuitivo.

Cada plano en formato DXF proporcionado por el STIG tiene una serie de capas como se muestra en la figura 5.4.2. Para generar un archivo *shapefile*, hay que abrir la utilidad *export*, y configurar las opciones de la figura 5.4.3. Es decir, los objetos se exportan como polígonos, se exportan todos los objetos del plano, y únicamente se exportan la capa `PUERTAS` y la capa *POLILINEAS_SUPERFICIES*, que contiene todas las habitaciones definidas como polígonos cerrados. Además en la pestaña "Data" se ha seleccionado el atributo `LAYER` que permitirá seleccionar objetos en las consultas SQL en función del tipo de capa, como se explicará más adelante.

A partir de este fichero *shapefile*, existen diferentes alternativas para introducirlo en nuestra base de datos PostGIS. La opción más sencilla consiste en utilizar el plugin *SPIT Shapefile to PostGIS Import Tool* del sistema de información geográfica *Quantum GIS*. Es un proceso sumamente intuitivo.

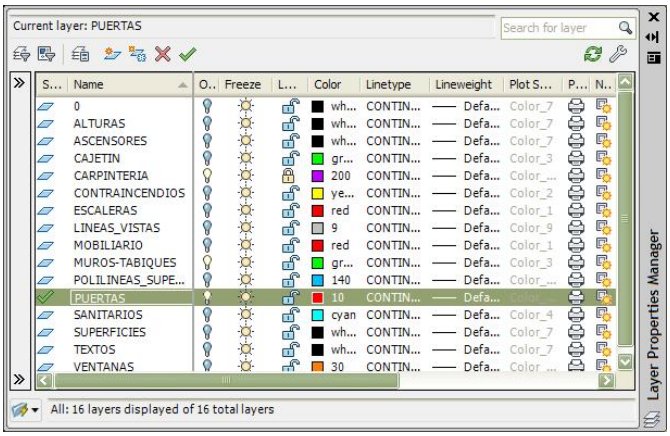


Figura 5.4.2: Capas existentes en un archivo DXF.

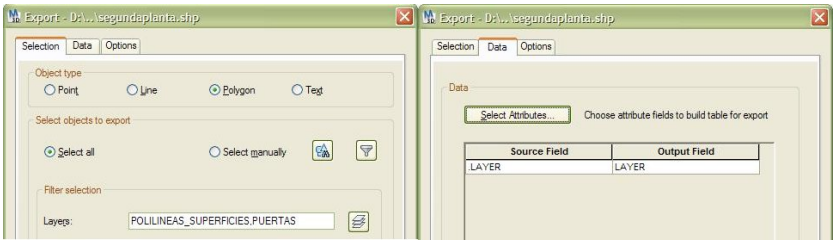


Figura 5.4.3: Opciones de exportación a formato shapefile.

5.4.2 FORMATO DE DATOS ESPACIALES

Para almacenar los datos espaciales en las bases de datos, la especificación OpenGIS [1] del OGC define dos formatos estándar que incluyen información acerca del tipo de objeto y de las coordenadas que lo definen:

- Formato *Well-known Text* (WKT): es un lenguaje de marcado para representar objetos geométricos en un sistema de referencia espacial. Está diseñado para ser fácilmente comprensible por el ser humano.
- Formato *Well-known Binary* (WKB): es un formato utilizado para transferir y almacenar un objeto geométrico como una secuencia continua de bytes, garantizando que la información sea exactamente la misma independientemente de la base de datos utilizada.

El formato WKT está definido utilizando la notación BNF *(Backus-Naur Form)*, donde se especifica la representación de los objetos geométricos que aparecen en la jerarquía de clases definida por el OGC de la figura 2.1.4: Geometry, Point, MultiPoint, Line, LineString, MultiLineString, LinearRing, Polygon, MultiPolygon, Triangle, Curve, MultiCurve, Surface, MultiSurface, PolyhedralSurface, Tin y GeometryCollection. En la tabla 5.4.1 se muestran algunos ejemplos concretos de la representación de diferentes geometrías.

5.4.3 ESQUEMA DE LA BASE DE DATOS

El estándar OpenGIS "Simple Features Specification for SQL" define un modelo de objetos geométricos (ver 2.1.4), las funciones necesarias para

Representación WKT	Tipo de Objeto
`POINT (10 10)`	Un punto
`LINESTRING (10 10, 20 20, 30 40)`	Una línea con 3 puntos
`POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))`	Un polígono de 4 vértices
`MULTIPOINT ((10,10), (20,20))`	Un multipunto formado por 2 puntos

Tabla 5.4.1: Ejemplos de objetos geométricos en formato WKT.

manipularlos (ver 2.1.6), así como un conjunto de tablas de metadatos para manejar la información espacial. Siguiendo las especificaciones de este estándar, el esquema conceptual de una base de datos espacial PostGIS se ilustra en la figura 5.4.4. y está formado por 3 tablas:

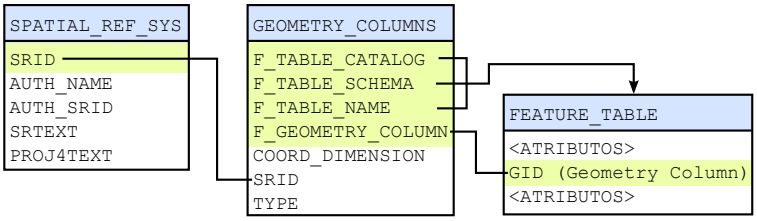


Figura 5.4.4: Esquema conceptual de la base de datos espacial PostGIS. Los campos en verde indican claves primarias.

- La tabla `SPATIAL_REF_SYS` describe el sistema de coordenadas y las transformaciones asociadas a cada objeto *Geometry*.
- La tabla `GEOMETRY_COLUMNS` se utiliza para saber qué tablas de la base de datos (`FEATURE_TABLE` en la figura 5.4.4) contienen información espacial con el tipo de datos *Geometry*.
- Cada tabla `FEATURE_TABLE` almacena una colección de características, donde se entiende por característica una abstracción de un objeto del mundo real.

5.4.3.1 *La tabla `SPATIAL_REF_SYS`*

Es una tabla de metadatos donde cada fila almacena la definición de un sistema de referencia espacial diferente, así como los detalles necesarios para efectuar transformaciones de un sistema a otro. En total PostGIS contiene cerca de 4000 sistemas de referencia espaciales, aunque se pueden definir otros no incluidos. Gracias a esta tabla se mantiene un sistema de coordenadas para todos los objetos geométricos almacenados en una tabla `FEATURE_TABLE`, de modo que las coordenadas de los vértices de cada objeto geométrico están asociadas a un lugar concreto del mundo. En la figura 5.4.5 se muestran algunas filas de ejemplo de la tabla `SPATIAL_REF_SYS` de nuestra base de datos PostGIS. La tabla está formada por cinco atributos:

- `SRID`: identificador del sistema de referencia espacial. Es la clave primaria de la tabla e identifica de manera unívoca a un sistema de referencia espacial en la base de datos.

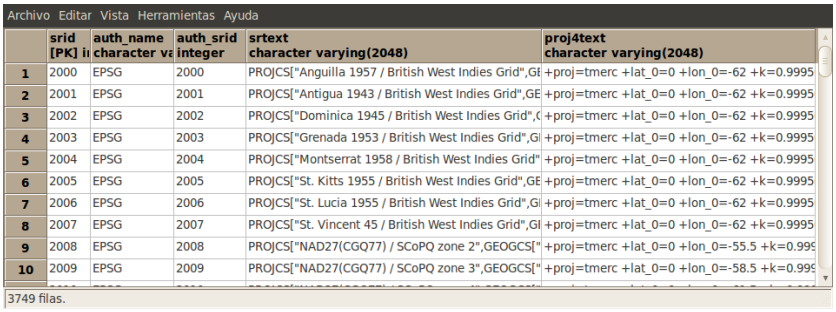


Figura 5.4.5: Tabla `SPATIAL_REF_SYS` de PostGIS

- `AUTH_NAME`: nombre de la autoridad del sistema de referencia espacial, que en la mayoría de los casos es el *European Petroleum Survey Group* (EPSG).
- `AUTH_SRID`: identificador del sistema de referencia espacial especificado por la autoridad `AUTH_NAME`. En PostGIS coincide con el atributo `SRID`.
- `SRTEXT`: descripción textual en formato WKT del sistema de referencia espacial.
- `PROJ4TEXT`: es un atributo propio de PostGIS que se utiliza para proporcionar transformaciones de coordenadas entre diferentes sistemas de referencia espacial utilizando la biblioteca Proj4.

5.4.3.2 *La tabla `GEOMETRY_COLUMNS`*

Es una tabla de metadatos que sirve como referencia para saber en qué tablas y atributos de la base de datos está almacenada la información espacial. Resulta de gran utilidad para los SIG puesto que les permite ofrecer al usuario final la posibilidad de elegir las tablas que contienen información espacial que puede ser visualizada en un mapa. La tabla está compuesta por ocho atributos, donde `OID` es un identificador de objeto usado internamente por PostGIS, y `F_TABLE_CATALOG`, `F_TABLE_SCHEMA`, `F_TABLE_NAME` y `F_GEOMETRY_COLUMN` forman una clave primaria compuesta. En PostgreSQL no existe el concepto de "catalog", así que este atributo se deja en blanco. Como esquema se utiliza por defecto "public". El resto de atributos tienen el siguiente significado:

- `F_TABLE_NAME`: se utiliza para indicar el nombre de la tabla que contiene una columna especial que almacena objetos de tipo geométrico.
- `F_GEOMETRY_COLUMN`: sirve para indicar qué columna de la tabla indicada en el campo `F_TABLE_NAME` es la que contiene la información geométrica de cada objeto.
- `COORD_DIMENSION`: es el número de dimensiones en el que están especificados los objetos geométricos. PostGIS soporta hasta cuatro dimensiones, donde la cuarta dimensión puede utilizarse como una dimensión temporal, por ejemplo.
- `SRID`: contiene el identificador del sistema de referencia espacial y hace referencia a la clave primaria de la tabla `SPATIAL_REF_SYS`.

- `TYPE`: contiene el tipo de objeto geométricos especificados según el Modelo de Objetos Geométrico del OGC (sección 2.1.4), por ejemplo `POINT`, `LINESTRING`, `POLYGON`, etc.

En la figura 5.4.6 se muestran algunas filas de ejemplo de la tabla `GEOMETRY_COLUMNS` de nuestra base de datos PostGIS. De acuerdo a esta figura, la tabla `segundaplanta` contiene la información geométrica de cada objeto en la columna `the_geom`, y los objetos almacenados son de tipo `MultiPolygon`.

Archivo Editar Vista Herramientas Ayuda

	oid	f_table_catalog [PK] character v	f_table_schema [PK] character v	f_table_name [PK] characte	f_geometry_column [PK] character varyi	coord_dimension integer	srid integer	type character varyi
1	17757	''	public	ciencias00	the_geom	2	-1	MULTILINESTRING
2	24584	''	public	ciencias00_poly	the_geom	2	-1	POLYGON
3	17740	''	public	ciencias02	the_geom	2	-1	MULTILINESTRING
4	17716	''	public	ciencias02poly	the_geom	2	-1	MULTIPOLYGON
5	24663	''	public	puertas	the_geom	2	-1	MULTIPOLYGON
6	24606	''	public	segunda	the_geom	2	-1	POLYGON
7	24647	''	public	segunda2	the_geom	2	-1	POLYGON
8	24678	''	public	segundaplanta	the_geom	2	-1	MULTIPOLYGON

Figura 5.4.6: Datos de ejemplo en la tabla `GEOMETRY_COLUMNS` de nuestra base de datos PostGIS.

5.4.3.3 *Tablas FEATURE_TABLE*

Las tablas de este tipo almacenan la información de tipo espacial, y existe una de estas tablas por cada fila que tenga la tabla `GEOMETRY_COLUMNS`. El único atributo obligatorio para esta tabla es la columna `GID` que es la clave primaria y contiene un identificador único para un objeto geométrico. El resto de atributos son opcionales, y se pueden definir a voluntad.

Por ejemplo, en la figura 5.4.7 se muestra la tabla `segundaplanta`, que aparece referenciada en la última fila de la figura 5.4.6. Como se puede observar, además del atributo `GID` obligatorio, la columna `THE_GEOM` almacena la información geométrica de cada objeto, que está codificada en un formato binario propio de PostGIS, y que es necesario transformar a formato WKT para poder interpretar en formato textual la información espacial. Además, se ha incluido la columna `LAYER` que indica el tipo de objeto que se representa, en este caso puertas. Se podrían incluir más atributos en esta tabla, pero en este caso no se ha estimado oportuno, dado que la información geométrica era más que suficiente.

Archivo Editar Vista Herramientas Ayuda

	gid [PK] serial	LAYER character varying(255)	the_geom geometry
1	1	PUERTAS	0103000000010000000500000A17091450AF74740
2	2	PUERTAS	0103000000010000000500000009899EDD4CC4C474C
3	3	PUERTAS	0103000000010000000500000003B0A65A299094840
4	4	PUERTAS	0103000000010000000500000003B0A65A299094840
5	5	PUERTAS	0103000000010000000500000004D0A65A299094840
6	6	PUERTAS	0103000000010000000500000003B0A65A299094840
7	7	PUERTAS	0103000000010000000500000703D5A59B8DE4640
8	8	PUERTAS	0103000000010000000500000CDCC16D4CC7C464C

Figura 5.4.7: Tabla `segundaplanta` con los objetos geométricos correspondientes a la segunda planta del edificio de la Facultad de Ciencias.

5.4.4 CONSULTAS POSTGIS PARA LA ASOCIACIÓN DE DATOS

El objetivo de esta sección es analizar cómo crear una consulta que permita obtener las coordenadas de los centroides de todas las puertas en un círculo de radio de R metros cuyo centro sea la posición estimada del robot (x, y). En los siguientes ejemplos se utilizará la tabla PostGIS correspondiente a la segunda planta del edificio de la Facultad de Ciencias de la Universidad de Salamanca (figura 5.4.7).

5.4.4.1 *Centroide de una puerta*

En primer lugar se necesita una consulta SQL que permita obtener las coordenadas del centroide de una puerta con un identificador de objeto geométrico (gid) concreto. En el código de la figura 5.4.8 se muestra esta consulta espacial. Como se ha explicado anteriormente, el

```
1  SELECT ST_AsText(ST_Centroid(the_geom))
2    FROM SEGUNDAPLANTA
3    WHERE gid = 5;
```

Figura 5.4.8: Consulta del centroide de la puerta con identificador 5.

atributo "the_geom" almacena la información del objeto geométrico. Las puertas están definidas como polígonos, y para obtener las coordenadas de su centroide se necesitan dos funciones: la función `ST_Centroid` , que devuelve las coordenadas del centro geométrico de un objeto, y la función `ST_AsText`, que transforma la definición de un objeto geométrico al formato WKT. Así, el resultado concreto de esta consulta es un punto en formato WKT:

```
POINT(48.1300010497545 -34.8961277236791)
```

5.4.4.2 *Puertas en un radio de distancia*

Una vez que se conoce cómo consultar el centroide de una puerta, el paso siguiente es consultar sólo aquellas puertas a una cierta distancia del robot. Para ello se utiliza la función `ST_DWithin`, que permite obtener aquellos objetos geométricos que estén a una determinada distancia de un objeto geométrico dado. Suponiendo que (x, y) sean los valores concretos de la posición estimada del robot, y R sea la distancia máxima de consulta de puertas R, la consulta espacial resultante se muestra en la figura 5.4.9.

```
1  SELECT ST_AsText(ST_Centroid(the_geom))
2    FROM segundaplanta
3    WHERE
4      ST_DWithin(the_geom,
5                 ST_GeomFromText('POINT(x y)'),
6                 R)
7    AND "LAYER" = 'PUERTAS';
```

Figura 5.4.9: Consulta para averiguar todas las puertas en un radio de ρ metros.

Como se puede observar, para poder especificar la posición del robot se necesita utilizar la función `ST_GeomFromText`, que construye un

objeto geométrico (en este caso un punto) a partir de su especificación en formato WKT. Además, se añade la restricción `"LAYER" = 'PUERTAS'` eliminando la necesidad de consultar tuplas correspondientes a otras capas de objetos existentes en la misma tabla. El resultado de la consulta es un conjunto de puntos en formato WKT, donde cada punto corresponde al centroide de una puerta.

CAPÍTULO 5.5

LOCALIZACIÓN MEDIANTE EKF

La aplicación de un filtro extendido de Kalman (EKF) en el caso de la localización de un robot móvil es ligeramente diferente a su aplicación en otros problemas de estimación, como el seguimiento de una cara en una imagen, por ejemplo, o incluso otros problemas más clásicos como el seguimiento de proyectiles. La diferencia fundamental reside en la ecuación de innovación, descrita en el apartado 3.2.3:

$$\tilde{\mathbf{z}}_{k+1} = \mathbf{z}_{k+1} - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) \tag{5.5.1}$$

Esta ecuación es muy simple en apariencia, pero se revela trascendental al aplicarla a la localización basada en características. Por una parte, las observaciones ($\mathbf{z}_{k+1}$) son el resultado de todo el proceso de detección de puertas (capítulos 5.2 y 5.3). Por otra parte, las predicciones de las observaciones ($\mathbf{h}(\hat{\mathbf{x}}_{k+1|k}) \equiv \hat{\mathbf{z}}_{k+1}$) son el resultado de la consulta al sistema de información geográfica (capítulo 5.4). Además, la dimensión de $\mathbf{z}_{k+1}$ es independiente de la de $\hat{\mathbf{z}}_{k+1}$. Esto significa que el número de puertas detectadas por el robot puede ser distinto al número de puertas consultadas al mapa. Esto da lugar al complicado problema de la *asociación de datos*, que no existe en el caso del seguimiento de un misil o una cara en una imagen, por ejemplo, puesto que las asociaciones entre predicciones y observaciones se conocen *a priori*.

El esquema de la figura 5.5.1 describe el funcionamiento general de nuestro sistema de localización. Mientras que en el capítulo 5.1 se

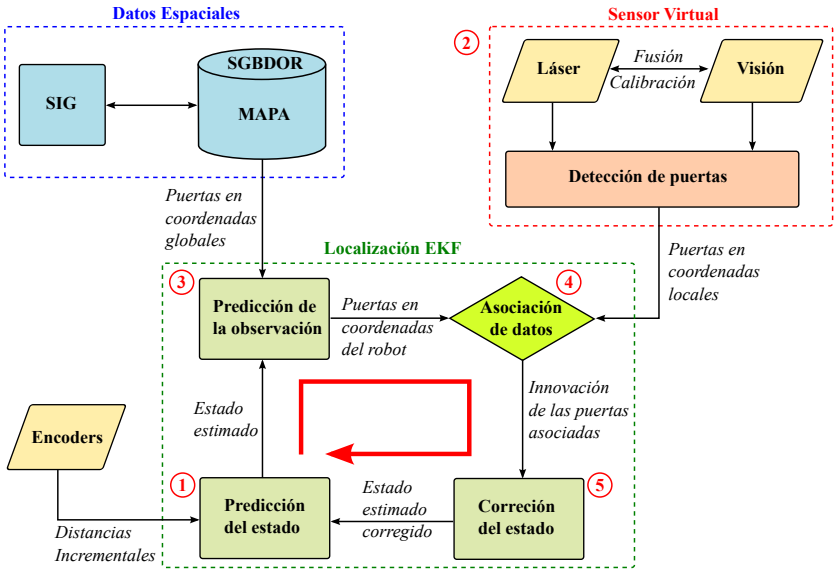


Figura 5.5.1: Esquema general de nuestro sistema de localización mediante EKF.

explicó la relación entre los componentes de línea punteada de este

mismo esquema, en las siguientes secciones se describen los cinco pasos principales del EKF en mayor detalle. La presentación está basada en el trabajo de *Durrant-Whyte et al.* [62] y *Siegwart et al.* [99].

5.5.1 PREDICCIÓN DE LA POSICIÓN DEL ROBOT

En el instante de tiempo k la posición estimada del robot es $\hat{\mathbf{x}}_{k|k}$ y la incertidumbre en la estimación se expresa mediante la matriz de covarianza $\mathbf{P}_{k|k}$ (ver apartado F.2). Exceptuando los casos en que el robot se encuentre detenido, su posición en el instante de tiempo $k+1$ será diferente de la anterior, de modo que su nueva posición $(\hat{\mathbf{x}}_{k+1|k})$ e incertidumbre $(\mathbf{P}_{k+1|k})$ deben ser recalculadas. El desplazamiento del robot durante el intervalo $[k, k+1]$ se debe a la acción de control $(\mathbf{u}_k)$, cuya medida se obtiene mediante los *encoders* instalados en el eje de cada rueda. Por lo tanto, se puede predecir la estimación de la posición del robot en el instante de tiempo $k+1$ en función de su ubicación en el instante k y su movimiento debido a la entrada de control:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{f}(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k) \tag{5.5.2}$$

En el caso de un robot diferencial, la función $\mathbf{f}(\cdot)$ se corresponde con el modelo odométrico del sistema, desarrollado en el apartado 3.3.3:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} \hat{x}_{k|k} \\ \hat{y}_{k|k} \\ \hat{\theta}_{k|k} \end{bmatrix} + \begin{bmatrix} -\Delta S_k \cdot \sin\left(\hat{\theta}_{k|k} + \Delta\theta_k/2\right) \\ \Delta S_k \cdot \cos\left(\hat{\theta}_{k|k} + \Delta\theta_k/2\right) \\ \Delta\theta_k \end{bmatrix} \tag{5.5.3}$$

Además de la nueva posición, es preciso predecir la incertidumbre asociada a la posición en el instante de tiempo $k+1$, que se calcula como:

$$\mathbf{P}_{k+1|k} = \nabla_x \mathbf{f} \cdot \mathbf{P}_{k|k} \cdot \nabla_x \mathbf{f}^T + \nabla_u \mathbf{f} \cdot \mathbf{Q}_k \cdot \nabla_u \mathbf{f}^T \tag{5.5.4}$$

y es el resultado de aplicar la ley de propagación de los errores (ver apendice E) a las dos variables de las que depende:

1. La matriz de covarianza $\mathbf{P}_{k|k}$: representa la incertidumbre asociada a la posición en el instante de tiempo anterior.
2. La matriz de covarianza $\mathbf{Q}_k$ (ver apartado 3.3.4): define la incertidumbre relacionada con la entrada de control, es decir, con el desplazamiento medido por los encoders.

En la ecuación (5.5.4) aparecen dos Jacobianos que preceden y suceden a cada una de estas dos variables, y su función es linealizarlas. Por una parte, el Jacobiano de la función de transición de estados $(\nabla \mathbf{f}_x)$ se debe evaluar en el estado $\mathbf{x} = \hat{\mathbf{x}}_{k|k}$, y se calcula como:

$$\nabla_x \mathbf{f} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x} & \frac{\partial \mathbf{f}}{\partial y} & \frac{\partial \mathbf{f}}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta S_k \cdot \cos(\hat{\theta}_{k|k} + \Delta\theta_k/2) \\ 0 & 1 & -\Delta S_k \cdot \sin(\hat{\theta}_{k|k} + \Delta\theta_k/2) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.5.5}$$

Por otra parte, el Jacobiano de la entrada de control ($\nabla_u \mathbf{f}$) se evalúa igualmente en el estado $\mathbf{x} = \hat{\mathbf{x}}_{k|k}$, y se calcula como:

$$\nabla_u \mathbf{f} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial \Delta S_d} & \frac{\partial \mathbf{f}}{\partial \Delta S_i} \end{bmatrix} = \begin{bmatrix} t_1 - t_2 & t_1 + t_2 \\ t_3 - t_4 & t_3 + t_4 \\ 1/b & -1/b \end{bmatrix} \tag{5.5.6}$$

donde b es la distancia entre las ruedas motrices, y las variables t1, t2, t3 y t4 toman los siguientes valores:

$$t_1 = -\frac{1}{2} \cdot \sin(\hat{\theta}_k + \frac{\Delta\theta_k}{2}) \tag{5.5.7}$$

$$t_2 = \frac{\Delta S_k}{2b} \cdot \cos(\hat{\theta}_k + \frac{\Delta\theta_k}{2}) \tag{5.5.8}$$

$$t_3 = \frac{1}{2} \cdot \cos(\hat{\theta}_k + \frac{\Delta\theta_k}{2}) \tag{5.5.9}$$

$$t_4 = \frac{\Delta S_k}{2b} \cdot \sin(\hat{\theta}_k + \frac{\Delta\theta_k}{2}) \tag{5.5.10}$$

5.5.2 OBSERVACIONES

El segundo paso consiste en que el robot observe y calcule la posición de un conjunto de características $\mathbf{z}_{k+1}$ en el instante de tiempo $k+1$. En este trabajo, las características utilizadas son puertas y se detectan por medio de un sensor virtual consistente en la fusión de datos láser y visión (capítulos 5.2 y 5.3). Por lo tanto, suponiendo que n_o es el número de características observadas, $\mathbf{z}_{k+1}$ es un conjunto de n_o elementos, donde cada elemento $\mathbf{z}_i$ es una observación independiente. Tal y como se especifica en el apartado 3.4.1, cada una de las observaciones contiene las coordenadas del centro de la puerta especificadas en el sistema de referencia *local* del robot:

$$\mathbf{z}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}^R \tag{5.5.11}$$

Después de adquirir las n_o observaciones en el instante de tiempo $k+1$, para cada una de ellas se necesita cuantificar el error en la observación por medio de una matriz de covarianza:

$$\mathbf{C}_i = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}^R \tag{5.5.12}$$

donde los elementos de la matriz especifican el error en los ejes del sistema de referencia cartesiano del robot y se calculan como se detalla en el apartado 3.4.4.

5.5.3 PREDICCIÓN DE LAS OBSERVACIONES

El objetivo de esta fase es predecir $\hat{\mathbf{z}}_{k+1}$, el conjunto de puertas visibles desde la posición del robot. Para ello se efectúa una consulta a la base de datos espacial del SIG (ver capítulo 5.4), utilizando como posición del robot la estimación odométrica calculada en el apartado 5.5.1: $\hat{\mathbf{x}}_{k+1|k}$. Suponiendo que la consulta a la base de datos predice n_p observaciones, entonces $\hat{\mathbf{z}}_{k+1}$ consiste en un conjunto de n_p elementos, donde cada

elemento $\hat{\mathbf{z}}_j$ es una predicción de una puerta, con las coordenadas de su centro geométrico especificadas en el sistema de referencia del robot:

$$\hat{\mathbf{z}}_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix}^R \tag{5.5.13}$$

No obstante, la base de datos espacial almacena las coordenadas de cada puerta en el sistema de referencia *global*. Por lo tanto, para poder asociar las características del mapa con las características detectadas por el robot, es necesario definir ambas en el mismo sistema de referencia. Para lograrlo, se convierten las coordenadas de cada puerta en el sistema de referencia global al sistema de referencia local del robot:

$$\mathbf{z}_j^G = \begin{bmatrix} x_j \\ y_j \end{bmatrix}^G \mapsto \mathbf{z}_j^R = \begin{bmatrix} x_j \\ y_j \end{bmatrix}^R \tag{5.5.14}$$

Así pues, para cada puerta $\mathbf{z}_j^G$ obtenida de la base de datos, se recalculan sus coordenadas suponiendo que la puerta es observada desde el punto de vista del robot, definido por su estado estimado $\hat{\mathbf{x}}_{k+1|k}$:

$$\begin{aligned} \hat{\mathbf{z}}_j &= \begin{bmatrix} x_j \\ y_j \end{bmatrix}^R = \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}, \mathbf{z}_j^G) \\ &= \begin{bmatrix} (x_j^G - \hat{x}_{k+1|k})\cos(\hat{\theta}_{k+1|k}) + (y_j^G - \hat{y}_{k+1|k})\sin(\hat{\theta}_{k+1|k}) \\ -(x_j^G - \hat{x}_{k+1|k})\sin(\hat{\theta}_{k+1|k}) + (y_j^G - \hat{y}_{k+1|k})\cos(\hat{\theta}_{k+1|k}) \end{bmatrix} \end{aligned} \tag{5.5.15}$$

donde $\mathbf{h}$ es la *función de observación* especificada en el apartado 3.4.3. Para cada predicción $\hat{\mathbf{z}}_j$ se necesita calcular su Jacobiano asociado, $\nabla_x \mathbf{h}_j$, evaluado en la última predicción del estado del robot $\mathbf{x} = \hat{\mathbf{x}}_{k+1|k}$:

$$\begin{aligned} \nabla_x \mathbf{h}_j = \left.\frac{\partial \mathbf{h}_j}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k+1|k}} &= \begin{bmatrix} \frac{\partial x_j}{\partial \hat{x}_{k+1|k}} & \frac{\partial x_j}{\partial \hat{y}_{k+1|k}} & \frac{\partial x_j}{\partial \hat{\theta}_{k+1|k}} \\ \frac{\partial y_j}{\partial \hat{x}_{k+1|k}} & \frac{\partial y_j}{\partial \hat{y}_{k+1|k}} & \frac{\partial y_j}{\partial \hat{\theta}_{k+1|k}} \end{bmatrix} \\ &= \begin{bmatrix} -\cos(\hat{\theta}_{k+1|k}) & -\sin(\hat{\theta}_{k+1|k}) & t_1 \\ \sin(\hat{\theta}_{k+1|k}) & -\cos(\hat{\theta}_{k+1|k}) & t_2 \end{bmatrix} \end{aligned} \tag{5.5.16}$$

donde los términos t_1 y t_2 se calculan como:

$$t_1 = (-x_j^G + \hat{x}_{k+1|k}) \cdot \sin(\hat{\theta}_{k+1|k}) + (y_j^G - \hat{y}_{k+1|k}) \cdot \cos(\hat{\theta}_{k+1|k}) \tag{5.5.17}$$

$$t_2 = (-x_j^G + \hat{x}_{k+1|k}) \cdot \cos(\hat{\theta}_{k+1|k}) - (y_j^G - \hat{y}_{k+1|k}) \cdot \sin(\hat{\theta}_{k+1|k}) \tag{5.5.18}$$

5.5.4 ASOCIACIÓN DE DATOS

Esta etapa es probablemente el aspecto más crítico de la localización basada en EKF. Hasta el momento, y gracias a las dos etapas anteriores, se dispone de:

- $\mathbf{z}_{k+1}$: un conjunto de n_o puertas obtenidas mediante el sensor virtual del robot y con sus coordenadas definidas en el sistema de referencia local del robot.
- $\hat{\mathbf{z}}_{k+1}$: un conjunto de n_p predicciones de puertas que el robot debería poder observar desde su posición estimada, obtenidas mediante una consulta a la base de datos espacial del SIG. Las coordenadas de cada puerta también están definidas en el sistema de referencia local del robot.

A partir de estos dos conjuntos de características en el mismo sistema de referencia, el problema de la asociación de datos consiste en averiguar la correspondencia entre cada puerta observada $(\mathbf{z}_i)$ y cada puerta consultada al mapa $(\hat{\mathbf{z}}_j)$. La dificultad del problema reside en que la correspondencia es desconocida *a priori*, y además el número de puertas observadas n_o y el número de puertas consultadas al mapa n_p normalmente no coincide.

5.5.4.1 *Algoritmo del vecino más próximo (nearest neighbor)*

Para resolver el problema de la asociación de datos existen diferentes algoritmos, que de acuerdo a Neira y Tardós [82] en general constan de dos elementos:

1. Una *prueba* para determinar la compatibilidad individual entre una observación $(\mathbf{z}_i)$ y una característica obtenida del mapa $(\hat{\mathbf{z}}_j)$.
2. Un *criterio de selección* que permita decidir cuáles son las mejores asociaciones de entre el conjunto de asociaciones compatibles.

Entre las diferentes opciones disponibles, se ha optado por utilizar el algoritmo del vecino más próximo *(nearest neighbor)* utilizando puertas de validación *(validation gates)* como prueba de compatibilidad individual. Las principales ventajas de este método con respecto a otros son su simplicidad conceptual y su reducida complejidad computacional.

Puertas de validación

Una *puerta de validación* define la máxima discrepancia tolerable entre una observación $(\mathbf{z}_i)$ y una predicción de una observación $(\hat{\mathbf{z}}_j)$. En otras palabras, es un mecanismo para rechazar asociaciones incorrectas y considerar solamente aquellas asociaciones en las que las puertas consultadas al mapa se encuentren a una distancia razonable de la puerta observada.

En el EKF, cada característica tiene asociada una matriz de covarianza para definir su modelo de error gaussiano. Las métricas euclídeas no se suelen emplear para calcular la distancia entre dos características porque no utilizan la información de la incertidumbre. Lo más habitual es utilizar como *puerta de validación* la *distancia de Mahalanobis*, que sí utiliza la incertidumbre de cada característica para calcular la distancia entre ambas. Para una observación $\mathbf{z}_i$ y una predicción $\hat{\mathbf{z}}_j$ la distancia de Mahalanobis se calcula como:

La distancia de Mahalanobis también se conoce en la bibliografía como 'innovación normalizada al cuadrado'.

$$M_{ij} = \tilde{\mathbf{z}}_{k+1}^{T}(i,j) \cdot \mathbf{S}_{k+1}^{-1}(i,j) \cdot \tilde{\mathbf{z}}_{k+1}(i,j) \tag{5.5.19}$$

Los términos que aparecen en la ecuación tienen el siguiente significado:

- $\tilde{\mathbf{z}}_{k+1}(i,j)$: es la *innovación en la medida*, es decir, la diferencia entre la observación i y la predicción j:

$$\tilde{\mathbf{z}}_{k+1}(i,j) = \mathbf{z}_i - \hat{\mathbf{z}}_j = \mathbf{z}_i - \mathbf{h}(\hat{\mathbf{x}}_{k+1|k}, \mathbf{z}_j^G) \tag{5.5.20}$$

- $\mathbf{S}_{k+1}(i,j)$ es la *covarianza de la innovación* y se calcula aplicando la ley de propagación de los errores (ver apéndice E) como:

$$\mathbf{S}_{k+1}(i,j) = \nabla_x \mathbf{h}_j \cdot \mathbf{P}_{k+1|k} \cdot \nabla_x \mathbf{h}_j^T + \mathbf{C}_i \tag{5.5.21}$$

  donde $\nabla_x \mathbf{h}_j$ se calcula según la ecuación (5.5.16) y $\mathbf{C}_i$ se calcula como (5.5.12).

Cuando las innovaciones en la medida tienen una distribución gaussiana, la distancia de Mahalanobis sigue una distribución χ^2 [9]. La forma de la distribución χ^2 depende de la dimensión del vector de innovación, que en este caso es bidimensional. Por lo tanto la *puerta de validación* se aplica por medio de la siguiente desigualdad:

$$M_{ij} < \gamma_n \tag{5.5.22}$$

donde n representa los grados de libertad de la distribución χ^2 y se corresponde con la dimensión de la innovación. En nuestro caso utilizamos γ_2. La integral de la distribución χ^2 desde 0 hasta γ_2 especifica la probabilidad de que la asociación entre i y j sea aceptada si $\mathbf{z}_i$ es realmente la observación que se corresponde con la predicción $\hat{\mathbf{z}}_j$. El valor de γ_2 se puede consultar en una tabla de probabilidad χ^2, y en nuestro caso hemos utilizado un valor del 95 % para las pruebas de compatibilidad entre características, lo que resulta en la siguiente *puerta de validación*:

$$\tilde{\mathbf{z}}_{k+1}^T(i,j) \cdot \mathbf{S}_{k+1}^{-1}(i,j) \cdot \tilde{\mathbf{z}}_{k+1}(i,j) < 5.991 \tag{5.5.23}$$

que aceptará el 95 % de asociaciones correctas, puesto que el 95 % de la probabilidad recae entre 0 y $\gamma_2 = 5.991$.

Vecino más próximo

La *puerta de validación* se aplica entre cada una de las n_o observaciones y cada una de las n_p predicciones. Cuando la distancia entre una observación $\mathbf{z}_i$ y una predicción $\hat{\mathbf{z}}_j$ es menor que γ_2 la asociación es aceptada por la *puerta de validación*. Aunque este es el caso más deseable, no es el más habitual, pudiendo suceder que:

- Una puerta detectada sea aceptada por la *puerta de validación* de varias puertas consultadas al mapa.
- Varias puertas detectadas sean aceptadas por la *puerta de validación* de una misma puerta del mapa.

En ambos casos existe incertidumbre con respecto a qué asociaciones serán las correctas. La solución es utilizar el criterio del vecino más próximo *(nearest neighbor)* para seleccionar las asociaciones que tengan el menor valor calculado en su puerta de validación. En otras palabras, se seleccionan como asociaciones correctas los valores M_{ij} más pequeños. Una opción poco recomendable para reducir el número de asociaciones incorrectas es utilizar una puerta de validación con un valor γ_2 menor. Esto incrementa la tasa de rechazo de asociaciones incorrectas, pero

al mismo tiempo presenta el inconveniente de que también se estarán rechazando asociaciones correctas [9].

Las observaciones que no cumplen una puerta de validación se ignoran en el proceso de localización. Tales observaciones son probablemente el resultado de objetos que no están realmente en el mapa y que han dado lugar a un falso positivo.

5.5.5 CORRECCIÓN DE LA POSICIÓN

El último paso del EKF consiste en aprovechar las asociaciones de características realizadas en el paso anterior para corregir la estimación de la posición del robot. Hasta el momento la mejor estimación de la posición disponible era $\hat{\mathbf{x}}_{k+1|k}$, correspondiente a la estimación en el instante de tiempo $k+1$ utilizando las observaciones disponibles hasta el instante de tiempo anterior, k. En esta etapa de corrección se utilizan las observaciones realizadas en el instante de tiempo $k+1$ para obtener la estimación del estado $\hat{\mathbf{x}}_{k+1|k+1}$ en el instante de tiempo $k+1$.

El primer paso para corregir la posición consiste en formar un vector con todas las observaciones $\mathbf{z}_i$ que han pasado alguna puerta de validación. Es decir, suponiendo que n puertas detectadas por el robot se han asociado satisfactoriamente con alguna puerta del mapa, se crea un vector $\mathbf{z}_{k+1} = \begin{bmatrix}\mathbf{z}_{i_1} & \cdots & \mathbf{z}_{i_n}\end{bmatrix}^T$. De la misma manera se procede con las puertas del mapa $(\hat{\mathbf{z}}_j)$ que se han utilizado en alguna asociación satisfactoria, dando lugar a un vector $\hat{\mathbf{z}}_{k+1} = \begin{bmatrix}\hat{\mathbf{z}}_{j_1} & \cdots & \hat{\mathbf{z}}_{j_n}\end{bmatrix}^T$. A partir de ambos vectores se calcula el vector de innovación $\tilde{\mathbf{z}}_{k+1}$:

$$\tilde{\mathbf{z}}_{k+1} = \begin{bmatrix}\mathbf{z}_{i_1} - \hat{\mathbf{z}}_{j_1} \\ \vdots \\ \mathbf{z}_{i_n} - \hat{\mathbf{z}}_{j_n}\end{bmatrix} \tag{5.5.24}$$

A continuación se apilan los Jacobianos $\nabla_x \mathbf{h}_j$ para cada una de las puertas del mapa que han sido utilizadas en alguna asociación, dando lugar al Jacobiano $\nabla_x \mathbf{h}$:

$$\nabla_x \mathbf{h} = \begin{bmatrix}\nabla_x \mathbf{h}_{j_1} \\ \vdots \\ \nabla_x \mathbf{h}_{j_n}\end{bmatrix} \tag{5.5.25}$$

Lo mismo se hace con las matrices de covarianza $\mathbf{C}_i$ para cada puerta detectada que se ha utilizado en alguna asociación, creando una matriz diagonal, donde el resto de los elementos de la matriz son 0:

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_{i_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \mathbf{C}_{i_n}\end{bmatrix} \tag{5.5.26}$$

A partir de estos valores se puede calcular la *covarianza de la innovación compuesta* como (5.5.27) y la *ganancia de Kalman* como (5.5.28).

$$\mathbf{S}_{k+1} = \nabla_x \mathbf{h} \cdot \mathbf{P}_{k+1|k} \cdot \nabla_x \mathbf{h}^T + \mathbf{C} \tag{5.5.27}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k} \cdot \nabla_x \mathbf{h}^T \cdot \mathbf{S}_{k+1}^{-1} \tag{5.5.28}$$

Finalmente, la corrección del estado y de su incertidumbre se calculan como:

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1} \cdot \tilde{\mathbf{z}}_{k+1} \tag{5.5.29}$$

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1} \cdot \mathbf{S}_{k+1} \cdot \mathbf{K}_{k+1}^{T} \tag{5.5.30}$$

CAPÍTULO 5.6

RESUMEN

En esta parte se han presentado una serie de capítulos correspondientes a nuestra propuesta de un sistema de localización basado en extracción de características de alto nivel y sistemas de información geográfica. En el capítulo 5.1 se presenta una perspectiva general de la relación entre los pasos principales que fundamentan el enfoque planteado, haciendo especial énfasis en los beneficios de la fusión sensorial a nivel de características entre el láser y la cámara.

En el capítulo 5.2 se detallan los algoritmos desarrollados para extraer las paredes de un pasillo en tiempo real utilizando únicamente el láser del robot. Básicamente la idea consiste en detectar la orientación del pasillo mediante un histograma angular, alinear el barrido láser con los ejes del sistema de referencia, y calcular la distancia a cada pared con un histograma-X. La detección de paredes en el espacio sensorial del láser es muy eficaz y mucho menos costosa computacionalmente que un análisis visual.

En el capítulo 5.3 se desarrollan los algoritmos para la detección de puertas en tiempo real. En primer lugar, y gracias a la fusión sensorial, se proyectan las paredes detectadas por láser en la imagen, construyendo regiones de interés donde buscar puertas. A continuación se utilizan características pseudo-Haar verticales para buscar las líneas correspondientes a los laterales de las puertas. Esta técnica se utiliza junto con la imagen integral para garantizar el procesamiento en tiempo real. Por último, se mide la distancia real (no en píxeles) entre pares de líneas verticales y se efectúa la clasificación final, obteniendo las coordenadas de las puertas desde el punto de vista del robot.

Para poder corregir la posición del robot, las puertas observadas se deben comparar con las puertas almacenadas en el mapa del entorno. Así, en el capítulo 5.4 se presenta un método para almacenar un mapa con información simbólica separada por capas en un sistema de información geográfica. Además se describe cómo realizar consultas espaciales a la base de datos espacial del SIG y obtener el conjunto de puertas visibles desde la posición estimada del robot.

Para concluir esta parte, el capítulo 5.5 combina las ideas desarrolladas en los 3 capítulos previos y describe su integración en un sistema de localización basado en un filtro de Kalman extendido. Se detalla la aplicación del modelo del sistema y el modelo de observación (desarrollados en la parte 3) en el ciclo predicción-corrección del EKF. Uno de los aspectos más importantes que se abordan en este capítulo es el problema de la *asociación de datos*, donde se explica la solución utilizada para hacer corresponder las n puertas observadas con las m puertas obtenidas del SIG, sin conocer las asociaciones correctas *a priori*.

Parte 6

RESULTADOS Y CONCLUSIONES

CAPÍTULO 6.1

RESULTADOS DE DETECCIÓN DE PUERTAS

Para evaluar el rendimiento de nuestro clasificador de puertas, hemos generado varios conjuntos de datos de prueba en diferentes ubicaciones de la Facultad de Ciencias de la Universidad de Salamanca. Para ello hemos utilizado nuestro robot *Morlaco* (ver apéndice. A), equipado con un láser SICK LMS 200 y una cámara Logitech Webcam Pro 9000. Ambos sensores están conectados a un ordenador portátil a bordo del robot con un procesador Intel Core i3 U380 con dos núcleos a 2.54 GHz, 2GB de RAM, y ejecutando Fedora 16 como sistema operativo. Todos los algoritmos han sido programados en lenguaje C.

6.1.1 PRUEBAS DE DETECCIÓN DE PUERTAS

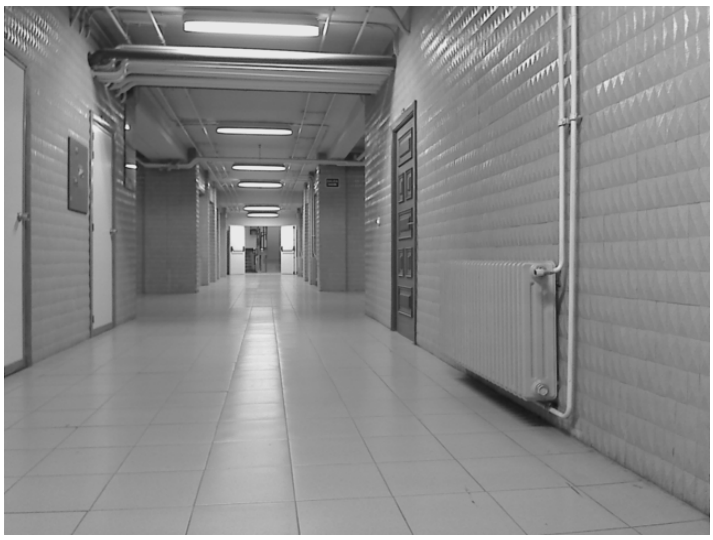
El conjunto de datos de prueba se ha obtenido mientras el robot navegaba por los pasillos del edificio de la Facultad de Ciencias de la Universidad de Salamanca. En total se han obtenido 500 imágenes con sus respectivos barridos láser. En la figura 6.1.1 se muestran cuatro entornos diferentes dentro del mismo edificio.

Por ejemplo, en la 6.1.1a se muestra una instantánea con puertas con marcos dobles en los dos lados del pasillo y algunos objetos como estanterías, radiadores y papeleras. En la figura 6.1.1b las puertas tienen marcos simples, debido a que el color de la puerta y el marco son exactamente iguales. Además hay objetos diferentes, como plantas

(a) Depto. de Informática y Automática

Figura 6.1.1: Diferentes entornos del conjunto de puertas de prueba.

(b) Secretaría de la facultad



(c) Pasillo principal de la facultad

(d) Departamento de Geología

Figura 6.1.1: Diferentes entornos del conjunto de puertas de prueba.

y bancos que impiden visualizar completamente las puertas. En la parte de la derecha hay varias ventanas que seguramente darían lugar a falsos positivos en el caso de utilizar detectores basados en el dintel de las puertas. La figura 6.1.1c muestra un pasillo más ancho con pequeños azulejos en las paredes que dan lugar a reflejos, y otro tipo diferente de puerta con elementos de relieve. Finalmente, en la figura 6.1.1d, se muestra otro entorno con puertas abiertas en la derecha y columnas y ventanas en la izquierda, además de mesas y estanterías.

En resumen, las imágenes tomadas incluyen diferentes tipos de puertas, distintos puntos de vista, y puertas totalmente visibles así como otras ocultadas parcialmente por objetos como radiadores, plantas, papeleras. Por ello, consideramos que los escenarios son lo suficientemente heterogéneos como para probar la eficacia de nuestro algoritmo de detección de puertas.

Además, son escenarios realistas sin ningún tipo de acondicionamiento previo, donde pueden aparecer ciertos problemas. Así, como la cámara está colocada a una altura aproximada de unos 75cm del suelo, un radiador o una papelera pueden dificultar bastante la visibilidad de una puerta. Otro problema es que en las puertas muy cercanas a la cámara el dintel no es visible. Por último, una dificultad fundamental es que cuanto más lejos del robot están las puertas, más difícil es detectarlas debido a que su anchura en la imagen se ve reducida a unos pocos píxeles, como se muestra en la figura 6.1.2. Por ejemplo,

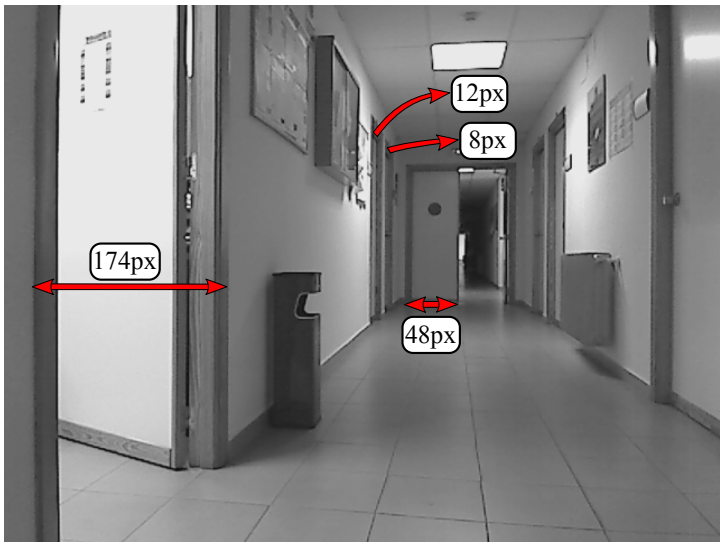


Figura 6.1.2: Diferencia de anchura en píxeles.

la primera puerta en la pared izquierda está a 1 metro de distancia y tiene una anchura de 174px, mientras que las siguientes puertas están a 6 y 7 metros de distancia y tienen una anchura de tan sólo 12px y 8px, respectivamente. A pesar de que la anchura real de las puertas es exactamente la misma, la reducción de anchura en píxeles es enorme debido a la proyección en perspectiva. Por esta razón, y para poder observar la influencia de la distancia en la eficacia del algoritmo, se han separado las pruebas en tres grupos en función de la distancia entre la puerta y el robot: i) hasta 3 metros, ii) de 3 a 5 metros, y iii) de 5 a 7

metros. Como medida de eficacia se ha utilizado la tasa de verdaderos positivos, que indica la cantidad de puertas detectadas correctamente:

$$V.Pos. = \frac{Num.Aciertos}{Num.Puertas} \tag{6.1.1}$$

así como la tasa de falsos positivos, que define la proporción de puertas inexistentes marcadas erróneamente como puertas reales:

$$F.Pos. = \frac{Num.Puert.Detectadas - Num.Aciertos}{Num.Puertas} \tag{6.1.2}$$

El resultado de nuestro método de clasificación de puertas se muestra en la tabla 6.1.1. De los resultados se observa que, como era de esperar,

Distancia	Num. Puertas	V. Pos. (%)	F. Pos. (%)
$d < 3$	278	91.01	2.16
$3 < d < 5$	348	83.91	5.17
$5 < d < 7$	296	57.09	6.08
Total	922	77.33	4.55

Tabla 6.1.1: Resultados de la clasificación de puertas

la tasa de verdaderos positivos disminuye a medida que la distancia a las puertas aumenta. Esto se debe principalmente a que la anchura de las puertas en la imagen disminuye con la distancia, pero también influye el hecho de que a mayor distancia el láser obtiene menos medidas. El análisis de los falsos positivos es también importante, y siempre es conveniente que la tasa sea lo mas baja posible, porque de lo contrario la localización del robot seguramente se vería afectada. Como se puede observar, la tasa de falsos positivos aumenta con la distancia, alcanzando un máximo de un 6 % de errores en el caso de las puertas más alejadas.

6.1.2 EJEMPLOS DE CLASIFICACIÓN DE PUERTAS

En esta sección se muestra gráficamente el rendimiento del algoritmo de detección de puertas propuesto. A continuación se muestran diferentes ejemplos de puertas correctamente clasificadas, así como fallos en la detección. En cada imagen aparece superpuesta la siguiente información:

- Líneas azules: delimitan las regiones de interés (ROI) de la imagen donde se esperan encontrar las puertas, y están calculadas a partir de la estimación de las paredes utilizando los datos del barrido láser. La intersección de estas líneas da como resultado el punto de fuga de la imagen.
- Puntos: representan la proyección en la imagen del barrido láser correspondiente. Se han utilizado tres colores diferentes para identificar los tres tramos de distancia especificados anteriormente: rojo (< 3 metros), verde (3 a 5 metros) y blanco (5 a 7 metros).
- Áreas verdes: indican la región exacta de la imagen en la que se ha detectado una puerta.

6.1.2.1 *Verdaderos positivos*

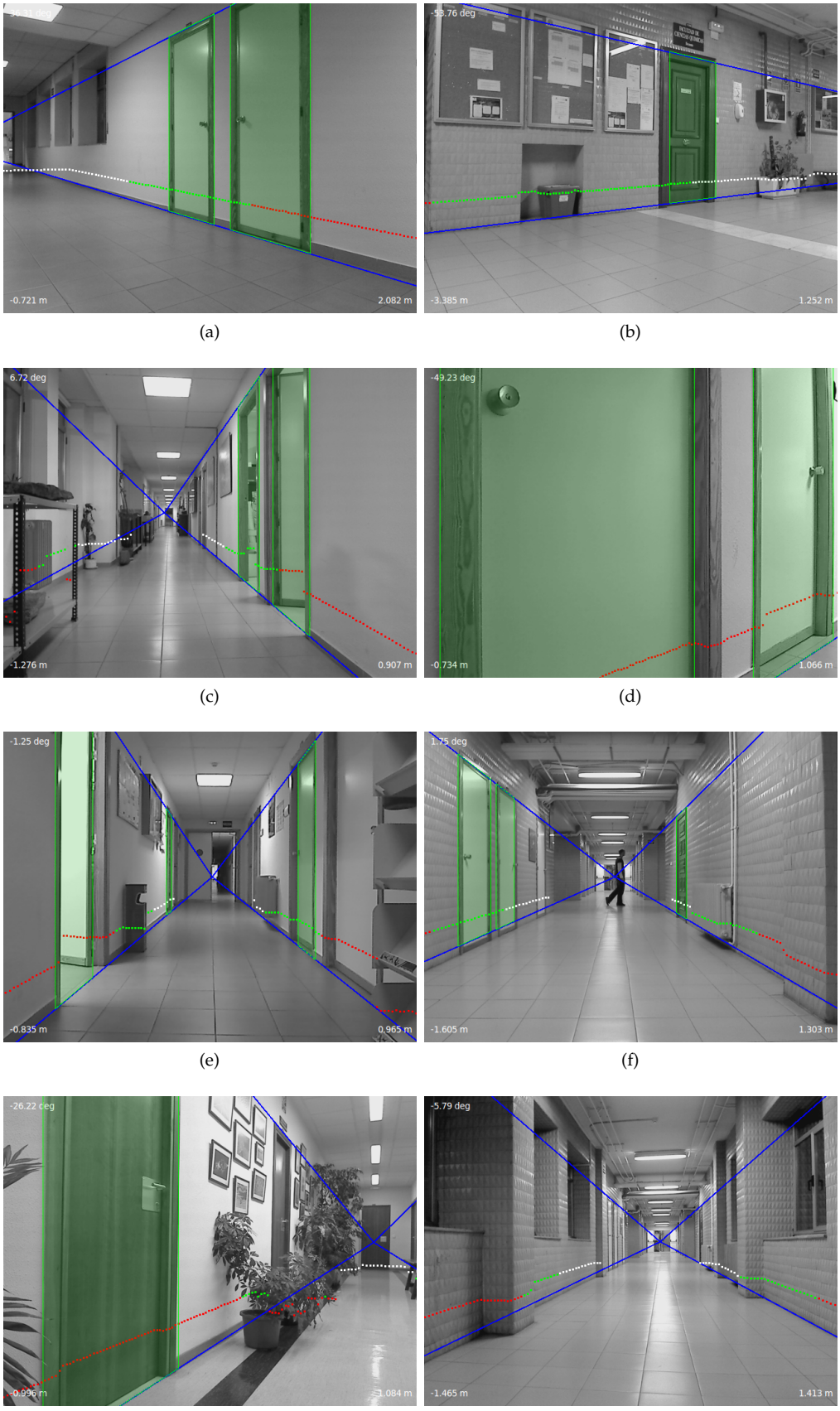
En la figura 6.1.3 se pueden observar algunos ejemplos de puertas detectadas correctamente.

La figura 6.1.3a muestra dos puertas totalmente alineadas con la pared. Al estar en el mismo plano que la pared, estas puertas serían totalmente invisibles para métodos basados únicamente en sónar o láser (ver sección 1.2.3).Además se pueden observar varias ventanas que podrían ser clasificadas erróneamente como puertas. En la figura 6.1.3b sucede lo mismo, los tablones de anuncios podrían dar lugar fácilmente a falsos positivos. El hueco en la pared bajo el tablón central también podría ser detectado como una puerta por un detector basado en sónar o láser. Nuestro método, sin embargo, gracias a la fusión sensorial calcula la altura esperada de cada puerta y las zonas de la imagen en las que deben encontrarse, eliminando incluso la posibilidad de detectar puertas reflejadas en el suelo. En la figura 6.1.3c se observa cómo el algoritmo funciona también con puertas completamente abiertas, mientras que en la figura 6.1.3d se puede apreciar la detección de dos puertas muy cerca del sensor, una de ellas ocupando más de la mitad de la imagen. En la figura 6.1.3e se muestran 3 puertas detectadas a diferentes distancias, incluyendo una a 7 metros y bastante difícil de observar dada su reducida anchura. En la pared derecha, una puerta similar no se ha podido detectar al estar parcialmente oculta por un radiador. Otro ejemplo de detección de puertas perfectamente alineadas con la pared se puede observar en la figura 6.1.3f, incluyendo una puerta en el rango de los 7 metros. En la figura 6.1.3g se muestra una puerta que ha sido correctamente detectada, y cómo la siguiente puerta no se ha identificado al estar parcialmente oculta por unas plantas. Finalmente, en la imagen 6.1.3h no se detecta ninguna falsa puerta en un escenario propenso a ello debido a ventanas y columnas, y donde las puertas más cercanas están en torno a 10 metros, y son prácticamente inapreciables.

6.1.2.2 *Falsos positivos*

Algunos ejemplos de falsos positivos se muestran en la figura 6.1.4. Por ejemplo, uno de los falsos positivos que se ha observado más frecuentemente es el de la figura 6.1.4a, en la pared derecha. En varias ubicaciones existen pares de puertas bastante cercanas donde los dos marcos contiguos se confunden con una puerta, dando lugar a un falso positivo. Esto se debe a que la anchura de la puerta más lejana se calcula defectuosamente por las limitaciones en la resolución angular del láser, mientras que la falsa puerta (más cercana) coincide con el ancho teórico definido para una puerta.

Otro de los falsos positivos más habituales es el que se produce en situaciones como la de la figura 6.1.4b. Normalmente se da entre el lateral real de una puerta y otro objeto que tenga la altura correspondiente a una puerta y se confunda con una línea vertical. En este caso se trata de una planta, pero también puede producirse con otros objetos, como estanterías o combinaciones de tablones y radiadores que coincidan formando una línea vertical debido a la perspectiva .



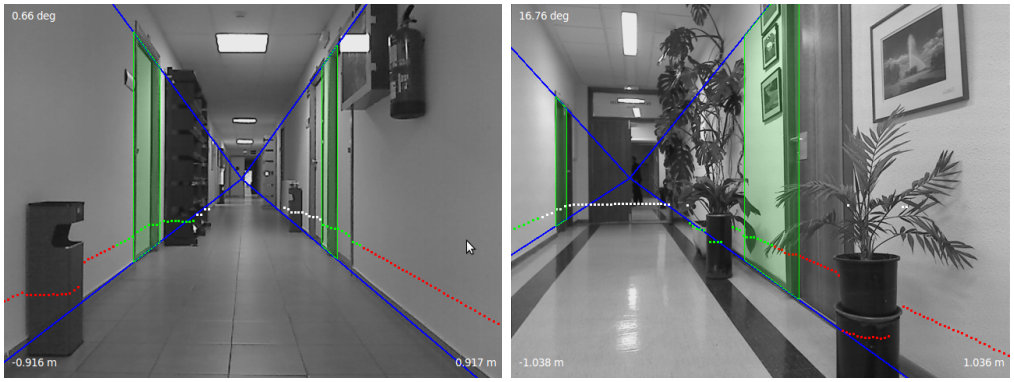
(a) (b)

(c) (d)

(e) (f)

(g) (h)

Figura 6.1.3: Detección de diferentes puertas. Los puntos corresponden a la proyección del láser en la imagen y su color indica la distanca: rojo - d $<$ 3m, verde - 3m $<$ d $<$ 5m, blanco - 5m $<$ d $<$ 7m.



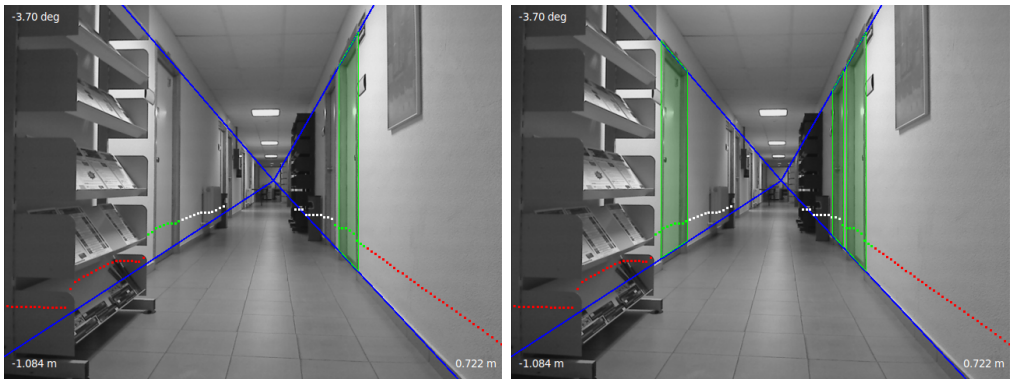
(a) Los marcos contiguos de dos puertas bastante próximas conducen a una clasificación errónea.

(b) Confusión de una planta con una línea vertical, provocando una detección incorrecta.

Figura 6.1.4: Falsos positivos

6.1.2.3 *Interpolación de puntos láser*

El papel del láser en la detección de puertas es fundamental, puesto que clasifica las puertas en función de la anchura medida entre pares de líneas verticales detectadas en la imagen. Sin embargo, a partir de una cierta distancia la resolución angular del láser es insuficiente y la medición del ancho de las puertas resulta menos precisa. La imagen 6.1.5a muestra el resultado de la detección de puertas sin interpolar las medidas del láser, fallando la identificación de dos puertas.



(a) Sin interpolación.

(b) Con interpolación.

Figura 6.1.5: Detección de puertas interpolando los puntos medidos por el láser.

Para mitigar este problema, se ha optado por interpolar las medidas del láser mediante un procedimiento de interpolación lineal. Cuando la anchura de las puertas se calcula a partir de la interpolación de los puntos del láser más cercanos a los extremos de la puerta, las medidas ganan en precisión y se consiguen detectar las dos puertas restantes, como se muestra en la imagen 6.1.5b.

6.1.3 ANÁLISIS DE TIEMPOS

Cualquier algoritmo basado en el análisis de imágenes tiene como reto conseguir que el tiempo de ejecución sea lo más reducido posible. Así,

para conseguir que un algoritmo se pueda ejecutar en tiempo real, es frecuente tener que recurrir a la utilización de imágenes de muy pequeño tamaño, como por ejemplo 320×240 px. El inconveniente de esta idea es que al reducir el tamaño de la imagen, se pierde información y es más difícil analizar los objetos distantes. Por el contrario, las imágenes de mayor resolución contienen también mayor información acerca del entorno. Además, hoy en día, cada vez son más habituales las cámaras capaces de proporcionar imágenes de alta definición en tiempo real (30fps), de modo que se necesitan algoritmos rápidos que puedan explotar las crecientes capacidades del hardware.

6.1.3.1 *Detección de puertas en tiempo real*

Para comprobar los tiempos de ejecución de nuestro algoritmo de detección de puertas, se han utilizado cuatro computadores diferentes como entornos de prueba, cuyas características hardware más importantes se muestran en la tabla 6.1.2. A bordo de nuestro robot *Morlaco* se encuentra un portátil Thinkpad Edge de Lenovo, de modo que sus tiempos de ejecución son considerados como la referencia principal,y aparecen marcados en negrita en todas las tablas.

Modelo	CPU (Intel)	Núcleos	RAM	Caché
○ Asus EeePC 901	Atom N270	1 (1.60 Ghz)	1GB	512 KB L2
○ Dell Inspiron 5150	Pentium D 920	2 (2.80 GHz)	3 GB	4 MB L2
● Dell	Core 2 Duo E8400	2 (3.00 GHz)	3.25 GB	6 MB L2
● Thinkpad Edge	**Core i3 U380**	**2 (2.54 GHz)**	**2GB**	**3MB L3**

○ Ubuntu 10.04 ● Fedora 16

Tabla 6.1.2: Características hardware de los entornos de prueba.

La cámara utilizada es capaz de obtener imágenes con una resolución de 640×480 px a 30 fps. Sin embargo, con su resolución máxima de 1600x1200 px, sólo puede capturar imágenes a 5 fps. Por ello, en las pruebas de localización del robot, la resolución elegida ha sido de 640×480 px, aunque en las pruebas de detección de puertas se han incluido imágenes de 1600×1200 px, y 3200×2400 px. Esto permite analizar hasta qué tamaño de imagen se podría ejecutar nuestro algoritmo en tiempo real. Los tiempos de ejecución del algoritmo de detección de puertas en los diferentes entornos de prueba, clasificados por computador y por tamaño de imagen, se muestran en la tabla 6.1.3. Cada valor de la tabla corresponde a un promedio de 100 ejecuciones.

La columna Total muestra el tiempo total de ejecución incluyendo el cálculo de la imagen integral. La columna Parcial representa el tiempo total menos el tiempo necesario para calcular la imagen integral. Se puede observar claramente como excluyendo el cálculo de la imagen integral, el tiempo de ejecución necesita un máximo de 4.50 ms en una imagen de 7 Mpx en un modesto EeePC 901. Esto da una idea del mínimo número de cálculos que se realizan, gracias a la fusión sensorial, el uso de la imagen integral, y la definición de regiones de interés en la imagen para reducir el área de búsqueda de puertas. En el resto de entornos de prueba, está claro que se podrían detectar puertas en tiempo real a cerca de 40 fps incluso para imágenes de 3200×2400 px. Concretamente, en el ordenador a bordo del robot, el tiempo total

	Total (ms)	Parcial(ms)	Tam. Img (px)
EeePC	7.6	1.51	640 × 480
	40.11	2.08	1600 × 1200
	158.63	4.50	3200 × 2400
Pent. D	1.39	0.45	640 × 480
	6.46	0.76	1600 × 1200
	23.62	1.41	3200 × 2400
C2 E8400	0.72	0.21	640 × 480
	6.40	0.77	1600 × 1200
	21.58	1.09	3200 × 2400
ThinkPad	**1.35**	**0.40**	**640 × 480**
	7.45	1.25	1600 × 1200
	26.59	1.70	3200 × 2400

Tabla 6.1.3: Tiempos de ejecución del algoritmo de detección de puertas en diferentes entornos de prueba. La columna *Parcial* indica el tiempo de ejecución sin incluir el tiempo de cálculo de la imagen integral.

para analizar completamente una imagen en busca de puertas es tan sólo de 1.35 ms ($\approx$ 740 fps) para imágenes de 640×480 px.

6.1.3.2 *Tiempo de ejecución de la imagen integral*

Dentro de las distintas partes de nuestro algoritmo de detección de puertas, el cálculo de la imagen integral es lo que necesita un mayor tiempo de ejecución. Dependiendo de la fórmula utilizada para calcular la imagen integral, el tiempo de procesamiento puede variar bastante, tal y como se muestra en la tabla 6.1.4.

La primera columna corresponde a la implementación de la ecuación (D.2), con un coste computacional aproximado de $3 \times w \times h$ operaciones, siendo w y h las dimensiones de la imagen. La segunda columna indica el tiempo de ejecución para el cálculo de la imagen integral según la ecuación (D.3), que tiene un coste aproximado de $2 \times w \times h$ operaciones. Obviamente, el tiempo de procesamiento debería ser teóricamente un 33 % menor en la segunda columna que en la primera. No obstante, esto sólo sucede para el portátil EeePC con un sólo núcleo. En el resto de entornos de prueba, la reducción de tiempo es aún mayor (hasta un 80 %) debido quizás a sus dos núcleos de procesamiento y un mayor tamaño de memoria caché. De estos resultados se puede deducir que la implementación más rápida de la imagen integral es un factor *crucial* y permite conseguir un ahorro de tiempo sustancial para poder ejecutar el algoritmo en tiempo real incluso en imágenes de gran tamaño.

6.1.3.3 *Comparación con la detección de líneas por convolución*

Una de las alternativas más populares para detectar bordes verticales en una imagen consiste en la utilización de un kernel de convolución adecuado. Algunos ejemplos son los operadores verticales de *Prewitt* o

	$3 \times w \times h(ms)$	$2 \times w \times h(ms)$	Mejora	Tam. Img (px)
EeePC	8.38	6.09	27.3 %	640×480
	52.38	38.03	27.4 %	1600×1200
	211.54	154.13	27.1 %	3200×2400
Pent. D	4.94	0.94	81.0 %	640×480
	32.04	5.70	82.2 %	1600×1200
	124.62	22.21	82.2 %	3200×2400
C2 E8400	1.96	0.51	74.0 %	640×480
	14.69	5.63	61.7.3 %	1600×1200
	57.50	20.49	64.4 %	3200×2400
ThinkPad	**4.09**	**0.95**	**76.8 %**	$\mathbf{640 \times 480}$
	27.85	6.20	77.7 %	1600×1200
	106.19	24.89	76.6 %	3200×2400

Tabla 6.1.4: Tiempos de procesamiento en el cálculo de la imagen integral, dependiendo del entorno de prueba y el tamaño de la imagen procesada.

el de *Sobel*, analizados en el apartado C.3. No obstante, la convolución es una operación costosa computacionalmente, y por ello se suelen utilizar kernels de dimensiones reducidas, ya que de lo contrario el tiempo de procesamiento sería demasiado elevado. Además, en la detección de líneas normalmente no se aplica una sola operación de convolución sino por lo menos 3: habitualmente un suavizado gaussiano más dos operadores de gradiente (uno vertical y otro horizontal).

En la tabla 6.1.5 se estudian los tiempos de procesamiento (en ms.) de diferentes implementaciones de convolución para las imágenes y los entornos de prueba anteriormente citados. Específicamente se analiza el resultado de aplicar los siguientes filtros de convolución de dimensiones 3×3:

1. Un kernel de convolución separable (ver apartado C.4), correspondiente a un operador de Prewitt vertical. Se representa en la tabla con la columna C.S. El coste computacional teórico es $N_{px} \times P \times Q$, siendo P y Q las dimensiones del kernel y N_{px} el número de píxeles de la imagen.

2. Un kernel de convolución no separable o estándar, con números aleatorios, identificado en la tabla como C.E. El coste computacional teórico es algo menor que en el caso anterior: $N_{px} \times (P + Q)$.

3. El mismo kernel separable que en el punto 1, pero implementado utilizando la librería de visión artificial OpenCV.

4. El kernel no separable del punto 2, implementado con OpenCV.

De los resultados se observa que efectivamente el cálculo de la convolución separable es mucho más rápido que el de la convolución estándar. En una imagen de 640×480 px, se podrían incluso aplicar varias operaciones de convolución separables y aún así se podrían procesar 30 fps. Sin embargo en un entorno tan modesto como es el EeePC,

	C.S.	C.E.	C.S. (OpCv)	C.E. (OpCv)	Tam. Img (Px)
EeePC	10.08	24.33	14.73	19.71	640 × 480
	62.33	151.84	89.95	123.01	1600 × 1200
	250.35	609.99	354.95	486.00	3200 × 2400
Pent. D	3.83	9.80	6.80	7.65	640 × 480
	22.80	50.70	35.94	40.03	1600 × 1200
	92.45	203.29	132.69	162.95	3200 × 2400
C2 E8400	3.46	3.73	3.94	4.31	640 × 480
	23.22	23.62	23.08	26.26	1600 × 1200
	94.01	95.71	89.36	104.96	3200 × 2400
ThinkPad	**2.98**	**8.52**	**10.52**	**11.77**	**640 × 480**
	18.32	53.30	59.56	73.35	1600 × 1200
	71.52	213.25	229.77	293.17	3200 × 2400

C.S. = Convolución Separable C.E. = Convolución Estándar

Tabla 6.1.5: Tiempos de ejecución (en ms) para un filtro de convolución 3×3, clasificados por tamaño de imagen y en entornos de prueba. Las pruebas incluyen convolución separable y convolución estándar, así como sus respectivas versiones en OpenCV.

probablemente no se podrían detectar líneas en tiempo real utilizando una convolución; posteriormente habría que ejecutar otras costosas tareas, como por ejemplo un algoritmo de componentes conectados (ver sección 5.3.3.1). La convolución estándar es más lenta, pero aún así se utiliza en ocasiones en casos donde no se puede separar la convolución. Un ejemplo es el filtro *Sigma* de Lee [61], que permite suavizar la imagen preservando los bordes. Otro aspecto a destacar es que en imágenes de gran tamaño, en todos los entornos probados, resultaría imposible ejecutar métodos de detección de puertas en tiempo real que utilizaran filtros de convolución.

Finalmente es importante comparar los resultados de la tabla 6.1.5 con los de la tabla 6.1.3. De esta comparación se observa que en una imagen de 640×480 px nuestro algoritmo de detección de puertas tarda 1.35 ms de media. Este tiempo da una idea de la rapidez de nuestro algoritmo, dado que es incluso menos de la mitad del tiempo que se tardaría en aplicar una única vez un filtro de Prewitt implementado mediante una convolución separable.

CAPÍTULO 6.2

RESULTADOS DE LOCALIZACIÓN

En este capítulo se exponen los resultados experimentales del sistema de localización propuesto en esta tesis. Está basado en la fusión de información odométrica con los datos procedentes del sistema de detección puertas (capítulos 5.2 y 5.3) y contrastados con el mapa almacenado en el SIG (capítulo 5.4). A la hora de probar el sistema de localización, es importante ejecutar los experimentos de manera autónoma, sin intervención humana, para evitar así influir en los resultados. Para facilitar esta labor se ha utilizado una integración de dos entornos de desarrollo de robots desarrollada en nuestro grupo de investigación. Tras explicar el sistema de construcción y ejecución de misiones para robots autónomos, se analizan los resultados experimentales.

6.2.1 ENTORNOS DE DESARROLLO ROBÓTICOS

Desde hace algo más de una decada, diferentes instituciones educativas han desarrollado entornos de desarrollo de robots, o RDEs, que sirven como ayuda a la hora de trabajar con robots autónomos y desarrollar nuevos algoritmos. Es un hecho que los RDE existentes difieren en muchos aspectos. En un exhaustivo estudio, Kramer y Scheutz [59] comparan las características, usabilidad, e impacto de uso (por número publicaciones científicas) de nueve RDEs de código abierto. Cada RDE tiene sus ventajas e inconvenientes, y dentro de las alternativas existentes, en GroUsal hemos optado por utilizar CARMEN y MISSIONLAB. Con el objetivo de poder aprovechar las mejores características de ambos, uno de los proyectos de nuestro grupo de robótica ha conseguido integrar satisfactoriamente CARMEN junto con MISSIONLAB, como se explicará a continuación. No obstante, antes de analizar los beneficios de dicha integración, se ofrece una concisa descripción de las características de estos dos RDE.

6.2.1.1 MISSIONLAB

MISSIONLAB [66] es un conjunto de herramientas software desarrollados por el Instituto Tecnológico de Georgia para facilitar el desarrollo y prueba de comportamientos tanto en robots individuales como en grupos de robots móviles. Está ideado para la construcción de misiones robóticas de estilo militar especificadas a un alto nivel de abstracción, e incluye software para controlar varios robots comerciales.

Uno de los puntos fuertes de este RDE es su usabilidad, evitando que los usuarios del sistema tengan que implementar cada misión a bajo nivel. Esto se consigue gracias al Editor de Configuraciones *CfgEdit* (*Configuration Editor*), una herramienta que permite construir misiones gráficamente a partir de comportamientos predefinidos, interconectán-

dolos mediante máquinas de estados finitos. Al compilar la misión gráfica, *CfgEdit* genera automáticamente el código fuente necesario y crea un *robot ejecutable* que se encarga de enviar las órdenes de la misión a un robot real. El proceso de compilación consta de tres fases, y se utilizan dos lenguajes intermedios denominados CDL y CNL:

$$\textit{CfgEdit} \rightarrow \text{CDL} \rightarrow \text{CNL} \rightarrow \text{C++}$$

El *robot ejecutable* generado mediante *CfgEdit* ejecuta la misión en un robot real gracias a que dispone de una biblioteca de funciones para comunicarse con el servidor hardware *HServer*. Este servidor define una interfaz de control abstracta para interactuar con los sensores y actuadores de los robots, y elimina la necesidad de generar código específico para un modelo de robot determinado. Otra importante característica de *HServer* es que permite estimar la posición de un robot fusionando diferentes fuentes de información sensorial [37]. La comunicación entre el *robot ejecutable* y *Hserver* se realiza por medio de la biblioteca de paso de mensajes distribuidos IPT [45], desarrollada por la Universidad de Carnegie Mellon.

6.2.1.2 CARMEN

CARMEN *es un acrónimo de* ***Car****negie* ***Mell****on Robot Navigation Toolkit.*

CARMEN [75, 76] es un RDE de código abierto desarrollado por la Universidad de Carnegie Mellon. Está preparado para trabajar con un sólo robot, y la modularidad es uno de sus principales principios de diseño. Proporciona módulos de comunicación con diferentes robots y sensores comerciales, almacenamiento y reproducción de *logs*, evitación de colisiones, localización, planificación de rutas, y construcción de mapas.

Los módulos de CARMEN están diseñados siguiendo una arquitectura de tres capas. La capa inferior proporciona interfaces abstractas para comunicarse con el hardware, efectúa los cálculos odométricos y planifica rotaciones simples y movimientos rectilíneos. La capa intermedia implementa tareas de navegación de más alto nivel, mientras que la capa de más alto nivel está reservada para tareas de usuario. El diálogo entre módulos confía en IPC [101], una biblioteca de comunicación interproceso desarrollada por la universidad de Carnegie Mellon. Esta biblioteca comparte bastantes similitudes con IPT (utilizada por MISSIONLAB), dado que ambas derivan de un antecesor común: el proyecto TCA *(Task Control Architecture)* [100].

6.2.1.3 *Integración de* CARMEN *y* MISSIONLAB

Uno de los problemas principales de MISSIONLAB es que no es compatible con versiones actuales de Linux, debido a problemas con las bibliotecas *Cthreads* [94] e IPT, puesto que ambas carecen de mantenimiento. Para solucionar este problema se ha sustituido *Cthreads* por *pthreads*, y se ha desarrollado *IPC_Adapter*, un nuevo componente para MISSIONLAB que transforma la interfaz de IPT en una interfaz compatible con IPC, utilizada por CARMEN y mantenida activamente.

Esta modificación de MISSIONLAB ha permitido además crear un nuevo modelo de comunicaciones entre ambos RDE basado en IPC, consiguiendo así que todos los servicios de MISSIONLAB sean utilizables por CARMEN y viceversa. Por ejemplo, todos los servicios de navegación de CARMEN están disponibles para MISSIONLAB, y se pueden utilizar

en el editor de misiones *CfgEdit* junto con otros comportamientos de MissionLab. Un beneficio añadido es que el sistema resultante sigue siendo multirobot.

A la hora de integrar los servicios de navegación de Carmen en MissionLab, se ha diseñado un comportamiento denominado Carmen_Navigate, definido en el lenguaje CNL de *MissionLab*. Este comportamiento recibe un par de coordenadas (x, y) como parámetros que le indican la posición que el usuario desea que alcance el robot. Para llegar a la posición objetivo, utiliza tanto el planificador de caminos como la evitación de colisiones de Carmen, y comprueba la posición utilizando nuestro sistema de localización, implementado en el módulo *doorLocalize*, tal y como se describe en la sección siguiente.

6.2.2 DISEÑO DE PRUEBAS DE LOCALIZACIÓN AUTÓNOMAS

El objetivo fundamental de este capítulo es verificar el funcionamiento de nuestro sistema de localización basado en la detección de puertas. Para realizar las pruebas experimentales de localización de manera totalmente autónoma con *Morlaco* (apénd. A), se ha utilizado la integración de Carmen y MissionLab descrita en el apartado anterior. Así, se logra que el robot tome todas las decisiones por sí mismo sin ningún tipo de intervención humana.

Para cada prueba de localización se ha creado una misión mediante el editor gráfico de misiones *CfgEdit* de MissionLab, utilizando el comportamiento Carmen_Navigate. A su vez este comportamiento basa su funcionamiento en una serie de módulos desarrollados para integrar *Morlaco* en Carmen:

- *morlaco*: un módulo que permite leer datos de los encoders de nuestro robot *Morlaco*, así como enviarle comandos de actuación a sus motores. Ambas tareas se llevan a cabo comunicándose con la controladora AX2850 de Roboteq, a bordo de *Morlaco*.

- *camera9000_pro*: aunque Carmen dispone de un módulo para leer imágenes de una webcam, no funcionaba con nuestro modelo de cámara. Nuestro nuevo módulo utiliza video4linux2 para obtener imágenes en formato YUYV de una Logitech Webcam 9000 Pro, leyendo únicamente el canal de luminancia, en blanco y negro y sin comprimir, que resulta la mejor opción para interpretar la imagen en busca de puertas.

- *doorDetection*: módulo de detección de puertas que depende directamente de los datos del láser y la cámara. Para acceder al laser Sick Lms 200 de *Morlaco* se utiliza el módulo *laser* de carmen, mientras que las imágenes se leen de nuestro módulo *camera9000_pro*.

- *doorDetectionGui*: interfaz gráfica para visualizar las puertas detectadas por *doorDetection*.

- *doorLocalize*: módulo de localización EKF que estima la posición del robot fusionando la información odométrica proporcionada por el módulo *morlaco* con el resultado de la asociación de datos entre las puertas detectadas por el módulo *doorDetection* y la consulta al mapa almacenado en el SIG.

- *doorLocalizeGui*: interfaz grafica para visualizar la asociación de datos, la incertidumbre y la posición del robot en tiempo real en un mapa almacenado en el SIG.

En la figura 6.2.1 se ilustran las relaciones entre los módulos creados, junto con el módulo *laser* ya existente en CARMEN y el nuevo comportamiento CARMEN_Navigate para MISSIONLAB. Todas las comunicaciones se realizan mediante la biblioteca de paso de mensajes IPC.

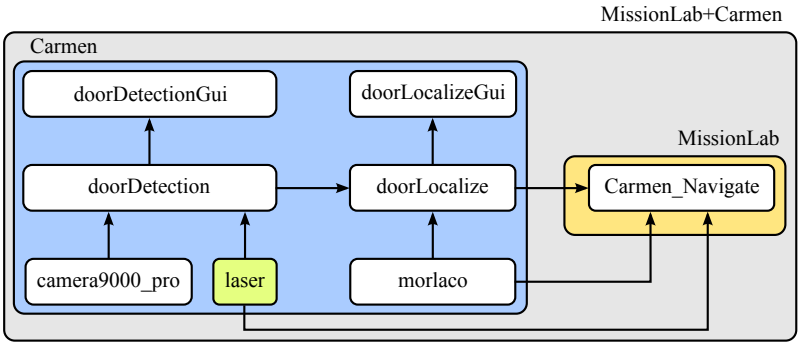


Figura 6.2.1: Intercomunicación (a través de IPC) de los módulos desarrollados para CARMEN y MISSIONLAB.

6.2.3 RESULTADOS EXPERIMENTALES

En entornos interiores, uno de los problemas habituales que afrontan los robots es mantenerse localizados en un pasillo. Esta dificultad se debe a que en ocasiones son entornos demasiado simples donde dos lugares diferentes pueden exactamente iguales, debido a la falta de objetos distintivos. En la figura 6.2.2 se muestra un escenario donde podrían aparecer problemas de localización.

El término habitual para referirse a estas situaciones es 'perceptual aliasing', que se podría traducir como 'similaridad perceptiva'.

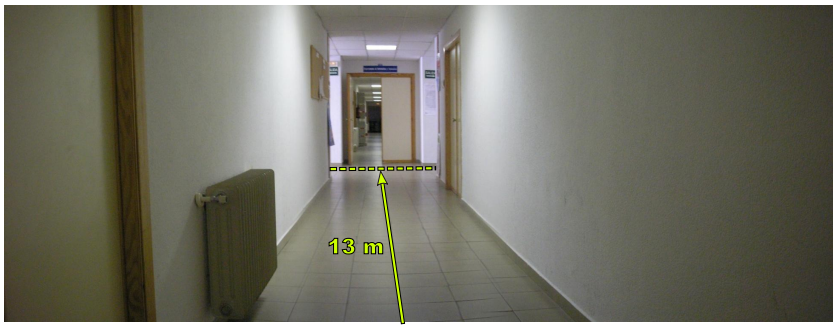


Figura 6.2.2: Falta de elementos distintivos en un pasillo de la facultad.

Si este pasillo es observado por un láser y se toman sucesivas capturas de datos a medida que el robot avanza en la dirección del pasillo, entonces la incertidumbre en la posición se puede mantener acotada en la dirección perpendicular al pasillo, gracias a las paredes. Sin embargo, el error de posición en la dirección del pasillo, es mucho más difícil de acotar porque para el láser todos los puntos de la paredes resultan similares. Una solución es detectar objetos salientes con respecto a la pared, como por ejemplo el radiador que se aprecia en la foto. Otra solución es que el final del pasillo esté dentro del alcance del láser (8m), aunque en este caso está a 13 metros y no es detectable. Este problema

aparece tanto utilizando datos láser en bruto, como sucede en el caso del localizador de CARMEN, como a un mayor nivel de abstracción, extrayendo las líneas correspondientes a las paredes.

La función natural de un pasillo es intercomunicar las distintas dependencias de un edificio, y de ahí que las puertas sean uno de los elementos más habituales en este tipo de entornos. Al detectar correctamente una puerta se restringe la incertidumbre tanto en la dirección del pasillo como en su perpendicular. Por consiguiente, nuestro sistema de localización debería funcionar correctamente en pasillos sin importar su longitud, como se demostrará en la siguiente prueba experimental.

6.2.3.1 *Prueba 1*

Para la primera prueba de localización se ha construido la misión que se puede observar en la parte superior de la figura 6.2.3 mediante el editor *CgfEdit* de MISSIONLAB, y transcurre en uno de los pasillos principales de la Facultad de Ciencias de la Universidad de Salamanca. Las

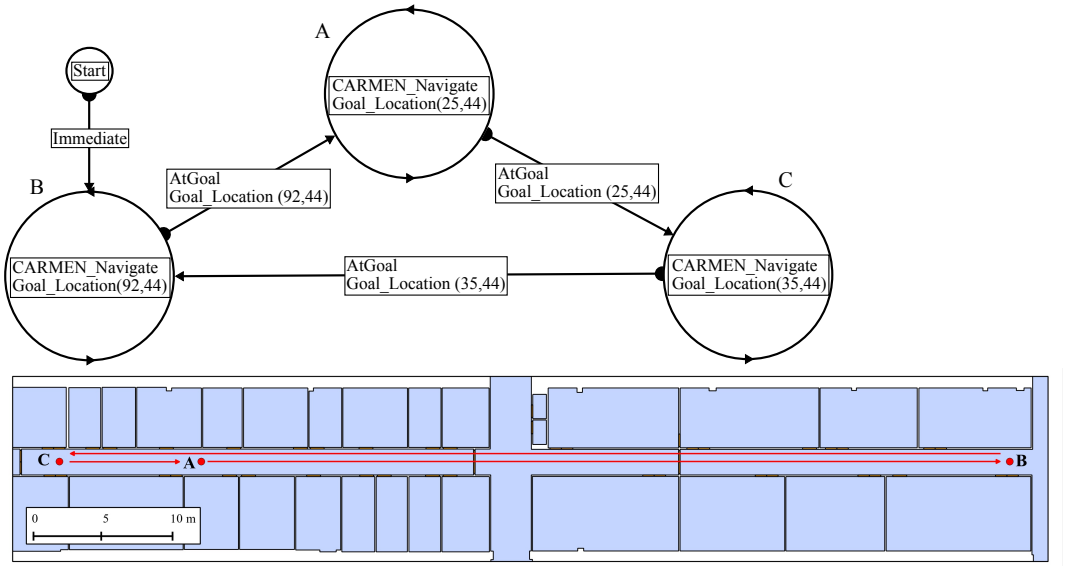


Figura 6.2.3: Misión en un pasillo de gran longitud ($\approx$ 70m de largo). Está programada para repetir indefinidamente el ciclo A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ A.

coordenadas que aparecen en cada comportamiento CARMEN_Navigate se corresponden con los puntos A, B y C del mapa que se muestra en la parte inferior de la misma figura. La distancia total de un ciclo completo A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ A es de unos 140 metros aproximadamente, y la misión se puede ejecutar indefinidamente al ser cíclica.

Una vez compilada la misión, gracias a la integración de los servicios de CARMEN en el comportamiento CARMEN_Navigate para MISSIONLAB, el robot funciona de manera totalmente autónoma. Así, se utiliza la planificación de caminos y la evitación de colisiones por láser de CARMEN. Al cruzarse personas en la ruta del robot, la trayectoria es automáticamente replanificada en caso de ser necesario. Una de las dificultades añadidas es que la iluminación del pasillo no es homogénea, al depender de fuentes de luz artificial a lo largo de toda su extensión. Esto provoca claroscuros como el de la figura 6.2.2, que incrementan la dificultad del análisis de la escena.

Después de repetir 8 ciclos completos, el robot se detuvo manualmente en el punto A, debido a que la batería del ordenador portátil estaba a punto de descargarse totalmente. El experimento duró aproximadamente 100 minutos, y el robot recorrió una distancia aproximada de 1.1 km. El resultado de la posición estimada de nuestro sistema de localización se puede observar en color verde en la figura 6.2.4, donde la etiqueta *"Final EKF"* corrobora que la estimación final de la posición coincide con el punto en el que el robot se detuvo.

La sorprendente ruta en rojo es el resultado de representar gráficamente la trayectoria del robot sin tener en cuenta las correcciones realizadas mediante la observación de puertas. Es decir, es el resultado de la integración odométrica, y la posición final estimada está a unos 50 metros de distancia con respecto al final real. A pesar de que existe un error sistemático evidente en la odometría, el *EKF* es capaz de corregirlo satisfactoriamente.

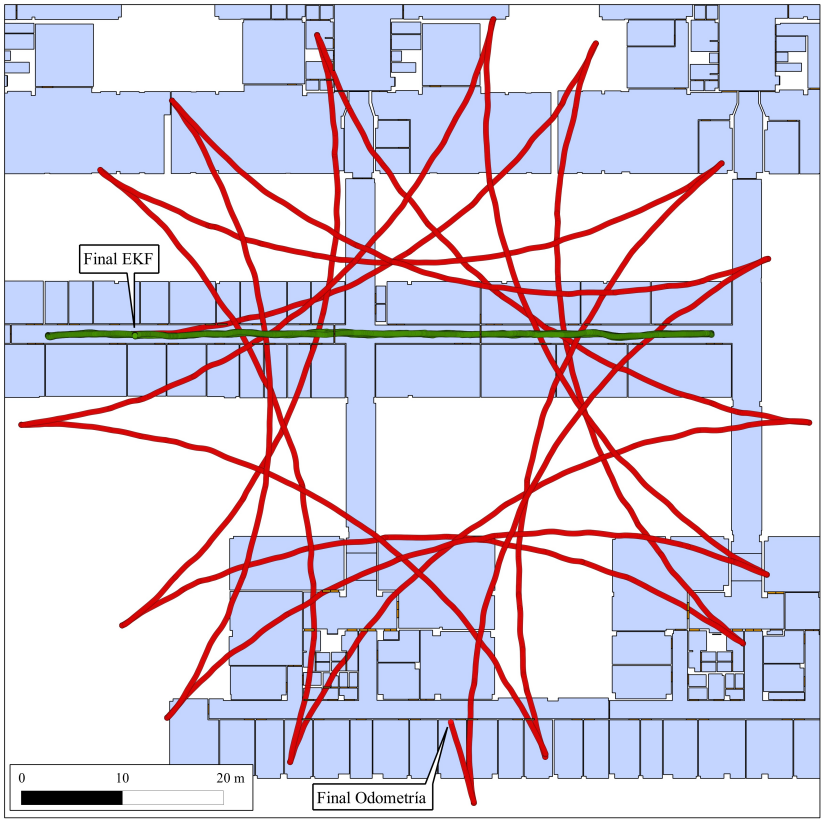


Figura 6.2.4: Trayectoria real seguida por el robot para realizar la misión 1. En rojo se muestra la ruta estimada que habría seguido el robot en caso de utilizar solamente la información odométrica. En verde, la ruta estimada utilizando la corrección con EKF por detección de puertas.

Análisis del error en la posición

La localización mediante EKF se basa en un modelo de distribución normal multivariante, lo que significa que cada elemento del vector de estado es una variable gaussiana. En nuestro caso el vector de estado tiene tres componentes: $\mathbf{x} = [x \quad y \quad \theta]^T$, que identifican la posición y orientación del robot en el sistema de referencia global. La matriz de

covarianza del estado recoge la varianza asociada a cada una de dichas variables:

$$\mathbf{P} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} \tag{6.2.1}$$

Al tratarse de variables gaussianas, el 95 % del error está contenido en el doble de la desviación estándar (2σ) y es el valor que se utilizará de aquí en adelante para representar el error en la posición estimada.

En la figura 6.2.5 se muestran los errores $2\sigma_x$, $2\sigma_y$ y $2\sigma_\theta$ a lo largo de la duración total de la misión descrita anteriormente.

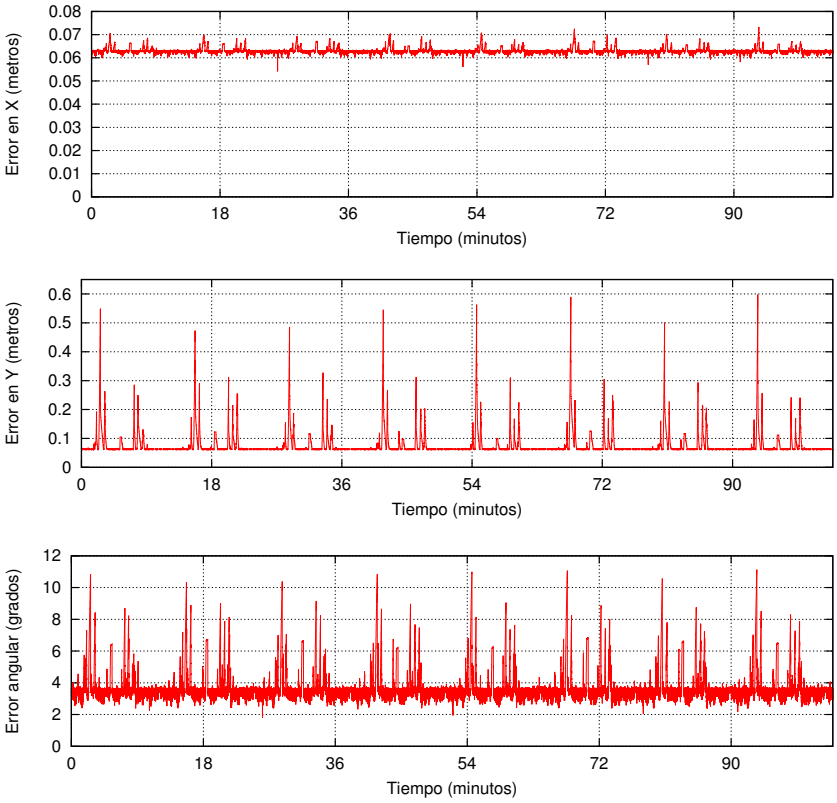


Figura 6.2.5: Errores $2\sigma_x$, $2\sigma_y$ y $2\sigma_\theta$ durante la misión de la figura 6.2.4.

Uno de los problemas encontrados a la hora de ejecutar el sistema de localización es que debido a la linealización que se realiza en el EKF, aparecían extraños errores en la matriz de covarianza, generándose valores negativos. Para resolver este problema, se ha optado por una compensación heurística, similar a la descrita en el apartado 3.2.4. En este caso se ha establecido un error mínimo $\sigma_x^2 = \sigma_y^2 = \sigma_{theta}^2 = 0.001$ en los términos de la diagonal principal, que equivale a 6.3 cm aproximadamente para $2\sigma_x$ y $2\sigma_y$, y 3.6º para $2\sigma_\theta$.

Es importante notar que las variables del vector de estado están definidas en el sistema de referencia global, y en este caso el eje X coincide con la dirección del pasillo. En la parte superior de la figura 6.2.5 se observa que el error en X está perfectamente acotado en torno al error mínimo establecido con pequeñas variaciones. El error en Y tiene mayores variaciones llegando a alcanzar 0.5 metros. Esto se debe a que hay tramos del pasillo donde no se ha podido detectar ninguna

puerta, y como es bien sabido, el error aumenta más rápidamente en la dirección perpendicular a la de avance. No obstante, nada más que se vuelven a detectar y asociar puertas correctamente, el error disminuye hasta el mínimo establecido. Lo mismo sucede para el error angular, con la salvedad de que los picos de mayor valor corresponden a los lugares en los que el robot efectúa giros de 180º.

Además de los errores para cada una de las variables de estado, en la figura 6.2.6 se muestra el área de la elipse que contiene un 95 % del error en las dimensiones $x - y$. Este valor es un mejor indicativo del error posicional al contener un error bidimensional. Como se puede observar, a lo largo de los más de 100 minutos de duración de la misión, el error alcanza un valor máximo de $0.2m^2$.

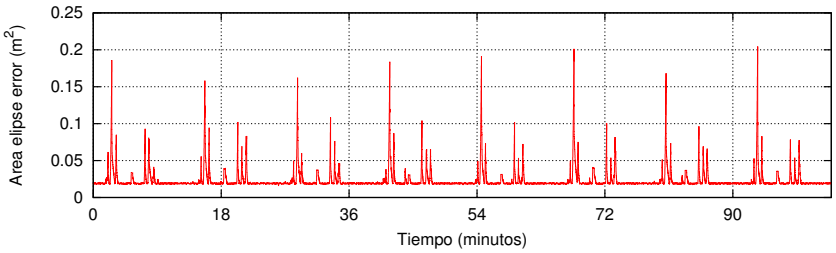


Figura 6.2.6: Área de la elipse de error en las dimensiones $x - y$ para un valor del 95 %.

6.2.3.2 *Prueba 2*

Para la segunda prueba de localización se ha construido la misión de la figura 6.2.3, ubicada en uno de los departamentos de Química Inorgánica de la Facultad de Ciencias. Las coordenadas que aparecen en cada comportamiento CARMEN_Navigate se corresponden con los puntos A, B, C y D del mapa que se muestra en la parte inferior de la misma figura. El robot parte del punto A, en el exterior del departamento, y acto seguido se dirige a la entrada del departamento (B). Una vez dentro, explora el entorno desplazándose hasta los puntos C y D, para regresar finalmente al punto B y repetir el ciclo BCDB hasta que la misión se detenga manualmente.

Después de aproximadamente 5 ciclos completos, el robot se detuvo manualmente en el punto C. El experimento duró aproximadamente 25 minutos, y el robot recorrió una distancia aproximada de 300 m. La ruta estimada de nuestro sistema de localización se muestra en color verde en la figura 6.2.8, y la etiqueta *"Final EKF"* indica que la posición final estimada coincide con el punto en el que el robot se detuvo realmente. La ruta en rojo corresponde a la ruta estimada utilizando únicamente la odometría, y la posición final estimada está a unos 8 metros de distancia con respecto al final real.

Análisis del error en la posición

En la figura 6.2.9 se muestra el error 2σ para las variables de estado x, y y θ, así como el área de la elipse de error correspondiente a un 95 %. Los valores representados en la gráfica, de acuerdo al modelo de error, indican que el robot está en la posición indicada con un 95 % de probabilidad. Al igual que en el experimento anterior, los errores están acotados la mayor parte del tiempo, salvo en circunstancias donde no

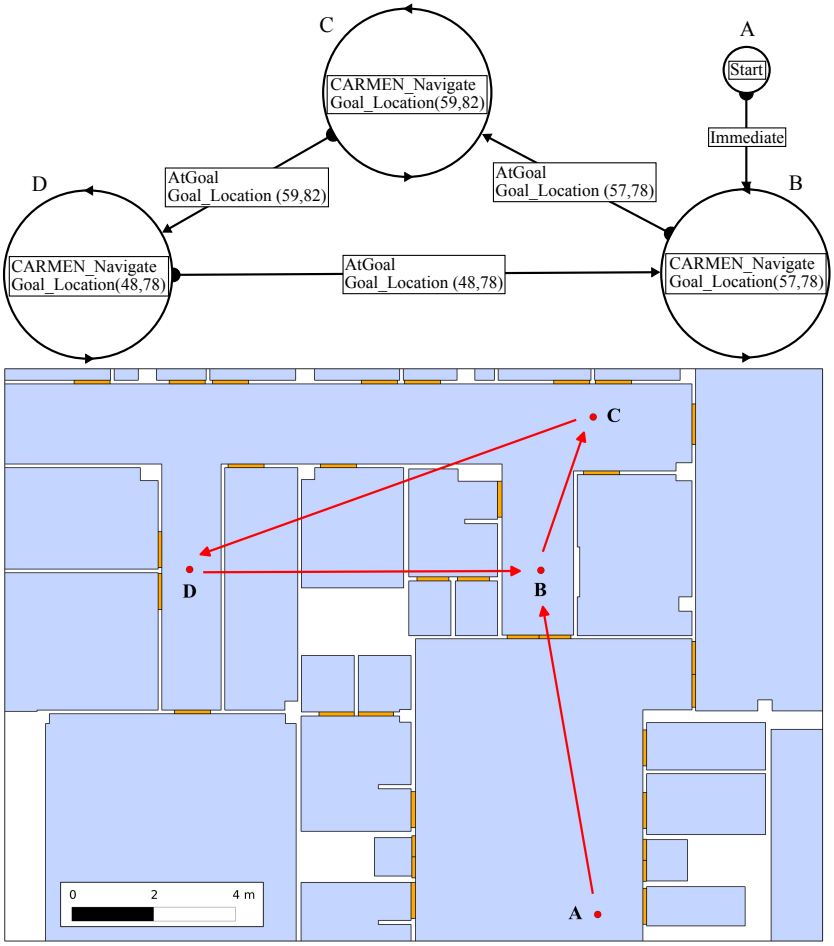


Figura 6.2.7: Misión en el interior de un departamento. La misión comienza en el punto A y a continuación define el ciclo BCDB, que se repite indefinidamente.

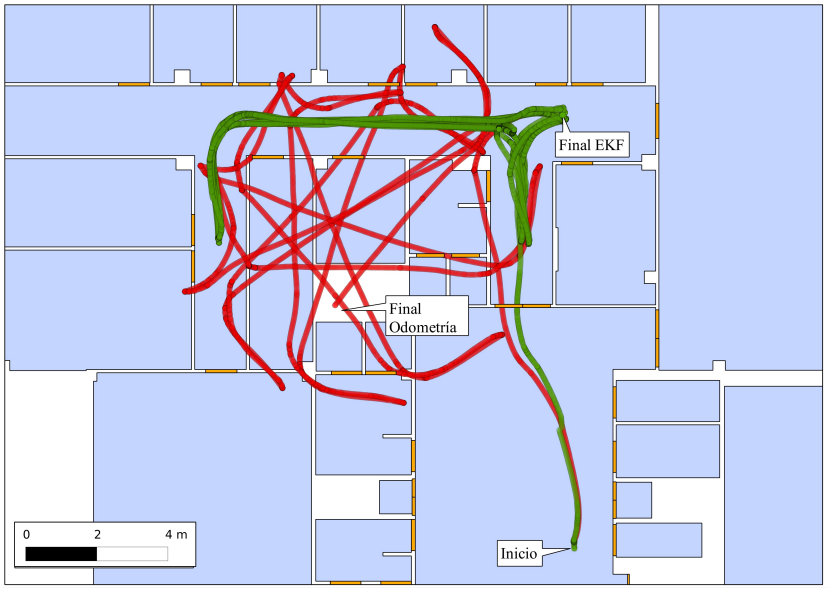


Figura 6.2.8: Trayectoria real seguida por el robot para realizar la misión 2. En rojo se muestra la ruta estimada seguida por el robot utilizando solamente la información odométrica. En verde, la ruta estimada utilizando la corrección con EKF por detección de puertas.

existen puertas que se puedan detectar. Los picos de error son cíclicos, lo que significa que el aumento en la incertidumbre se produce en las mismas zonas del entorno.

El error en X está acotado en torno al error mínimo establecido llegando a alcanzar un valor máximo de 30 cm. La ausencia de picos de error en Y en torno a los 5 y 10 minutos, indica que en el segundo y tercer ciclo el robot consiguió detectar puertas y mantener el error al mínimo, pero en el resto de ciclos no fue capaz de detectar esas mismas puertas. El error angular aumenta principalmente en las posiciones B y D, donde se producen giros de 180º y el planificador autónomo tomó la decisión de que el robot girase dando la espalda a las puertas. No obstante, al volver a asociar puertas correctamente, el error disminuye hasta el mínimo establecido.

Finalmente la gráfica inferior de la figura 6.2.9 muestra el área de la elipse que contiene un 95 % del error en las dimensiones $x - y$. A lo largo de los 25 minutos de duración de la misión, el error alcanza un valor máximo de $0.14m^2$, y durante la mayor parte de la misión el error se sitúa en un mínimo de $0.02m^2$.

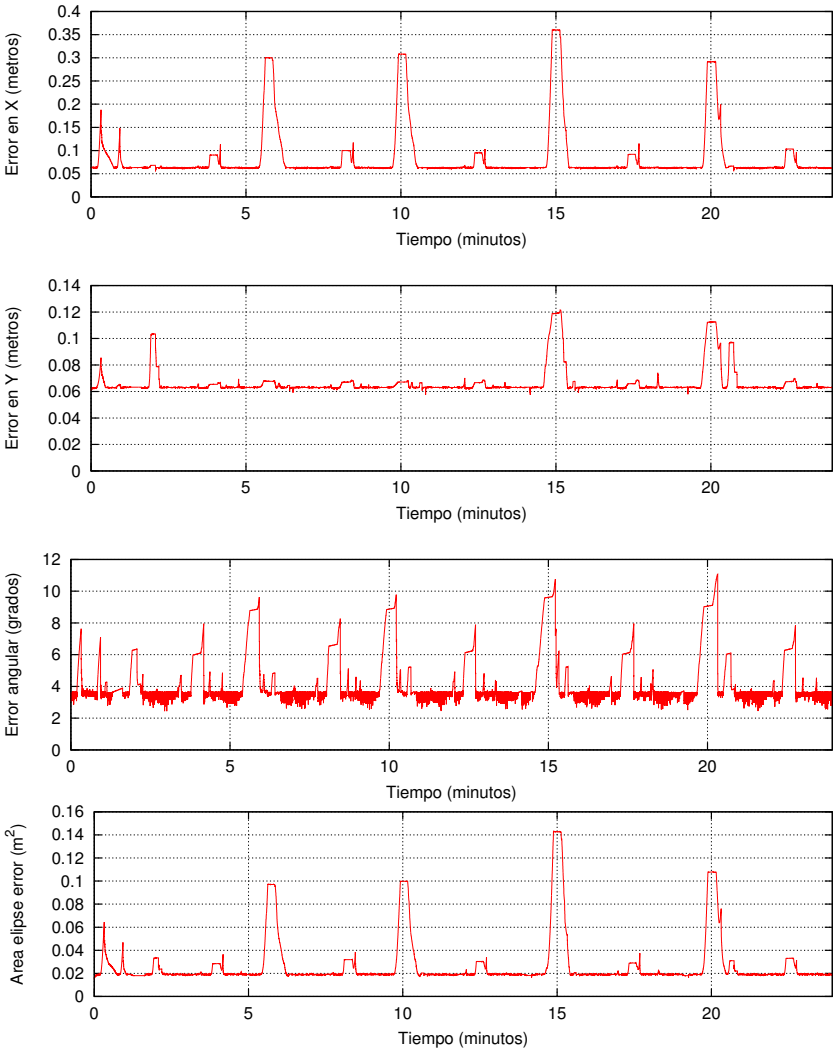


Figura 6.2.9: Errores $2\sigma_x$, $2\sigma_y$ y $2\sigma_\theta$ y área de la ellipse de error del 95 % durante la trayectoria estimada mediante EKF de la figura 6.2.8.

CAPÍTULO 6.3

CONCLUSIONES Y TRABAJO FUTURO

La finalización de una tesis doctoral no sólo supone una oportunidad para mirar hacia atrás y observar desde una perspectiva global el trabajo realizado, sino que también constituye una ocasión para concebir nuevas ideas relacionadas con la investigación llevada a cabo y desarrollarlas en el futuro.

6.3.1 CONCLUSIONES GENERALES

Desde que la palabra *robot* fue introducida en la literatura por el escritor *Karel Čapek* en 1921, la mayor parte de la sociedad imagina un futuro donde los robots son máquinas conscientes, con inteligencia y habilidades físicas similares o superiores a las humanas, y capaces de desenvolverse sin problemas en cualquier entorno humano. Algunas de estas visiones son ya una realidad, puesto que desde hace décadas existen robots que nos superan en velocidad, precisión, fuerza o resistencia. No obstante, a pesar de esta superioridad en ciertas tareas físicas, los robots están aún muy lejos de igualar nuestras habilidades intelectuales y cognitivas. Esto les impide ser capaces de desarrollar tareas similares a las que cualquier persona puede realizar en el interior de los edificios, que es donde el ser humano pasa la mayor parte de su tiempo.

La razón fundamental de que un robot se encuentre presente en el interior de un edificio es que pueda ayudar al ser humano en cualquier tipo de tareas: entretenimiento, seguridad, limpieza, ayuda personal a mayores o discapacitados, etc. Además, existen dos razones por las que es muy importante que los robots mantengan una estimación fiable de su posición en un edificio. La primera es que la localización resulta imprescindible para navegar de forma autónoma. La segunda es que existen determinados sistemas en los que el robot se considera un recurso de actuación, y como tal, el sistema necesita mantenerlo localizado. Dos campos donde se desarrollan este tipo de sistemas son la inteligencia ambiental y la gestión de inmuebles *(facility management)*. El propósito principal de un sistema de inteligencia ambiental es mejorar un edificio para ayudar al ser humano de forma inteligente y transparente, y para conseguir este fin los robots constituyen un recurso de actuación esencial. En los sistemas de gestión de inmuebles, el objetivo es optimizar los recursos (espaciales, humanos, etc.) de uno o varios edificios de gran tamaño, como centros educativos u hospitales, por ejemplo. En este caso un robot se podría encargar de la limpieza, la seguridad del edificio, u otras tareas, y se consideraría como un recurso robótico.

Tomando como base estas ideas, se ha desarrollado un sistema de localización que permite determinar la ubicación de un robot en el interior de un edificio en todo momento. A pesar de que este problema

ha sido estudiado con anterioridad, existen dos diferencias principales que hacen que nuestra propuesta sea innovadora:

- La estimación de la posición está basada en la extracción de información simbólica (características de alto nivel) del entorno en tiempo real.
- Se ha integrado nuestro sistema de localización junto con un *Sistema de Información Geográfica* (SIG).

Además de estas dos ideas, uno de los aspectos más relevantes y que más ha influido en los resultados de la investigación ha sido sin lugar a dudas el uso de técnicas de fusión multisensorial. Concretamente, la fusión se ha realizado a tres niveles de abstracción diferentes:

- Fusión a bajo nivel: se ha utilizado un método de calibración ya existente para combinar los datos en bruto del sensor láser y el sensor de visión y expresarlos en un sistema de referencia común.
- Fusión a nivel intermedio: se ha desarrollado un método para extraer características en el espacio sensorial del láser y definir regiones de interés en las imágenes obtenidas por la cámara, facilitando así el proceso de extracción de información simbólica.
- Fusión a alto nivel: se ha utilizado un filtro extendido de Kalman (EKF) para estimar la posición del robot a partir de diferentes fuentes sensoriales, fusionando información odométrica con la extracción de características mediante láser y visión.

6.3.2 CONCLUSIONES ESPECÍFICAS

En este apartado se describen conclusiones más detalladas en relación a la extracción de información simbólica, los sistemas de información geográfica, el sistema de localización y el desarrollo de misiones de navegación autónomas.

6.3.2.1 *Extracción de características de alto nivel en tiempo real*

En el interior de un edificio se puede extraer información simbólica detectando características de alto nivel. De entre los diferentes objetos existentes, las puertas son uno de las más frecuentes y más regularmente repartidos por el entorno, y por ello se han utilizado como elemento clave para la localización de nuestro robot móvil.

El sistema de detección de puertas que se ha desarrollado está fundamentado en la construcción de un sensor virtual, fusionando información multisensorial proporcionada por el láser y la cámara de visión a bordo del robot. A bajo nivel, los datos del láser son proyectados en las imágenes de la cámara, añadiendo así información de profundidad a la imagen. A nivel intermedio, se detectan las paredes en el espacio sensorial del láser y se proyectan en la imagen. La detección de paredes con láser es muy eficaz y mucho menos costosa que un análisis visual. Gracias a esto, las paredes en la imagen sirven como regiones de interés, consiguiendo reducir el espacio de búsqueda de las puertas y evitando búsquedas innecesarias en el techo o en el suelo.

Para minimizar el tiempo de procesamiento se han utilizado características pseudo-Haar verticales para encontrar las líneas correspondientes a los laterales de las puertas. Esta técnica se utiliza junto con la

imagen integral, que permite sumar y restar áreas de píxeles mediante un número extremadamente reducido de operaciones. Gracias a la fusión de datos a bajo nivel, se mide la distancia real (no en píxeles) entre los pares de líneas verticales detectadas, y se efectúa la clasificación final, almacenando la ubicación de las puertas detectadas. Todo el proceso es extremadamente rápido, superando muy ampliamente los requisitos de tiempo real necesarios para el sistema de localización. Por poner un ejemplo, incluso con imágenes de algo más de 7 Mpx se ha obtenido una velocidad de casi 40 fps.

El sistema propuesto ha sido probado en escenarios no preparados o modificados previamente, y con un amplio conjunto de puertas (con diferente color, forma y acabado), bajo diferentes condiciones de iluminación y puntos de vista, lográndose un alto porcentaje de aciertos con una baja tasa de falsos positivos. Además, el sistema funciona independientemente de que las puertas estén abiertas o cerradas, e incluso aunque las puertas estén perfectamente alineadas con respecto a la pared.

6.3.2.2 *Sistemas de Información Geográfica*

La Universidad de Salamanca dispone del *Servicio Transfronterizo de Información Geográfica* (STIG), que entre otras tareas se encarga de mantener planos actualizados de todos y cada uno de los edificios de la universidad. La información espacial modelada incluye tanto datos métricos del entorno como información simbólica (puertas, ventanas, ascensores, etc) con la ventaja de estar separada por capas.

Al integrar nuestro sistema de localización con un SIG, se ha podido reutilizar muy fácilmente la información espacial modelada por el STIG, gracias a la interoperabilidad de los SIG con otros sistemas. De este modo no sólo se evita tener que construir un mapa desde cero, sino que los datos quedan almacenados en una base de datos espacial PostGIS.

El robot interactúa con la base de datos espacial del SIG después de detectar un conjunto de puertas en el entorno por medio de su sensor virtual. La consulta consiste en obtener las puertas del mapa que son observables desde su posición estimada mediante odometría. La información obtenida es fundamental para el EKF y permite corregir la estimación de la posición del robot confiando en el proceso de asociación de datos.

Una ventaja de los SIG es que están específicamente diseñados para ser utilizados por el ser humano, de forma que la información que almacenan se puede visualizar y manejar por cualquier persona. Además, los datos se pueden georreferenciar, lo que significa que están identificados de forma única en el espacio. Esto permite añadir nuevos mapas de otros edificios y calcular rutas entre ambos, por ejemplo. Incluso se podrían añadir datos de otras ciudades u otros paises, y gracias a la escalabilidad del sistema y a su base de datos espacial, no sería necesario modificar la estructura del sistema, ni tampoco disminuiría su rendimiento.

6.3.2.3 *Localización basada en características de alto nivel*

El sistema de localización desarrollado está basado en la aplicación de un ciclo predicción-corrección mediante un EKF. La fase de predicción estima la posición del robot empleando únicamente la información

odométrica de sus encoders. La fase de corrección es más compleja y se basa en la asociación de las características de alto nivel detectadas por el robot con las mismas características almacenadas en el SIG. Este proceso de asociación de datos es muy complejo porque la correlación es desconocida *a priori*, y se ha resuelto mediante un algoritmo del vecino más próximo con puertas de validación basadas en la distancia de Mahalanobis.

Cabe destacar que tanto los aspectos teóricos del EKF como su implementación práctica han representado un gran desafío para el autor de este trabajo. Además, se ha observado que un aspecto esencial para el correcto funcionamiento del EKF consiste en modelar adecuadamente la incertidumbre del sistema y la de las observaciones.

6.3.2.4 *Misiones de navegación autónomas*

El funcionamiento del sistema de localización propuesto se ha comprobado mediante la ejecución de un conjunto de misiones totalmente autónomas en escenarios reales de la Facultad de Ciencias de la Universidad de Salamanca.

Para poder lograr un nivel total de autonomía, se han desarrollado módulos para el control del robot a bajo nivel, para la detección de puertas y para la localización, así como módulos para visualizar la posición del robot, las puertas detectadas y la incertidumbre de ambos. Todos estos módulos se han incluido en un entorno de desarrollo robótico (RDE) desarrollado en GroUsal y que consiste en la integración de CARMEN y MISSIONLAB. Gracias a este RDE integrado se ha podido verificar el funcionamiento satisfactorio del sistema de localización desarrollado, diseñando las misiones mediante MISSIONLAB y utilizando CARMEN para la planificación de caminos.

En una misma misión, nuestro robot ha sido capaz de recorrer más de 1 km de distancia durante casi 2 horas de duración y ha logrado mantener una estimación de la posición acotada durante toda la misión. Durante las misiones no se ha efectuado ningún tipo de intervención humana, y se han realizando tareas de evitación de obstáculos en caso de aparecer personas en la trayectoria del robot.

Los resultados de localización han demostrado la robustez del enfoque propuesto, al mismo tiempo que han permitido detectar los puntos débiles del sistema, e idear posibles líneas de trabajo futuro para mejorarlo.

6.3.3 TRABAJO FUTURO

A partir de la conclusiones extraídas, se pueden esbozar algunas ideas para mejorar y extender las capacidades del sistema actual. Una primera idea sería utilizar modelos más sofisticados para la asociación de datos, aprovechando la correlación existente entre las observaciones. Esto permitiría aumentar la robustez del sistema de localización y disminuir las posibilidades de divergencia del EKF.

Otro aspecto interesante sería extender la funcionalidad del sistema de extracción de información simbólica, que por el momento solamente está preparado para detectar puertas. En algunos puntos del entorno esto puede presentar problemas al no existir puertas en el campo de visión del robot. Para mejorar el sistema en este aspecto, una posibilidad sería utilizar las paredes detectadas por el láser, mientras que otra

opción sería desarrollar sistemas de detección de características de alto nivel adicionales, capaces de observar ventanas o radiadores, que también son objetos muy presentes en la mayoría de edificios.

En relación a la capacidad sensorial y de actuación del robot sería deseable añadir un brazo robótico para posibilitar la apertura de puertas, o incluir nuevos sensores como unidades de medida inercial, con el fin ampliar las funcionalidades del robot.

Con respecto a los sistemas de información geográfica, gracias a su base de datos espacial se podrían añadir nuevos objetos al mapa a medida que fuesen detectados, y completar así el modelo del mundo. Además, se podrían aprovechar las capacidades de visualización y compatibilidad con el ser humano de los SIG para diseñar un sistema que permita a un usuario final construir misiones de manera gráfica.

Finalmente, cabe destacar que se están realizando dos proyectos de investigación en GroUsal relacionados con la localización relativa basada en la odometría visual, así como la localización global mediante filtros de partículas y la extracción de puertas. Ambos proyectos se integrarán en el futuro con los resultados presentados en esta tesis doctoral, completando las funcionalidades del sistema actual.

Parte 7

APÉNDICES

APÉNDICE A

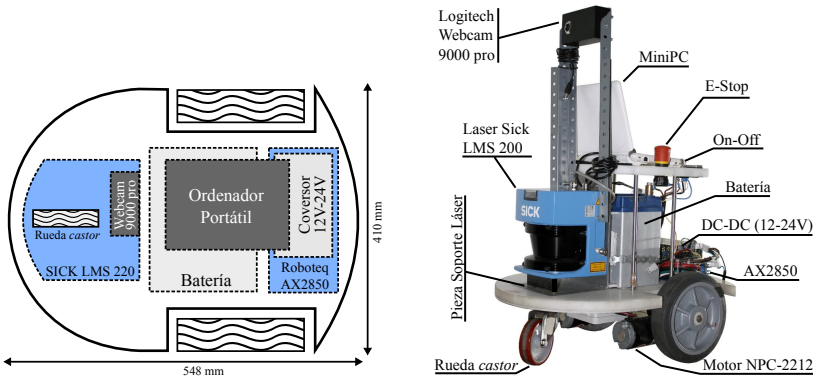
ROBOT MÓVIL MORLACO

Para poder comprobar de manera empírica en el mundo real el funcionamiento del sistema de localización presentado en esta tesis doctoral, se ha diseñado el robot móvil autónomo *Morlaco* en el Grupo de Robótica de la Universidad de Salamanca. El montaje mecánico ha sido realizado prácticamente en su totalidad por la empresa *Nacho y Cuervo S.L.*, con sede en Guijuelo, mientras que en GroUsal hemos realizado algunas pequeñas modificaciones. En este apéndice se describe brevemente el diseño físico, el sistema sensorial y de actuación, así como el sistema de control y los módulos software desarrollados para comunicarse con los dispositivos del robot.

A.1 DISEÑO

El primer paso en la fase de diseño mecánico del robot ha sido decidir la configuración de las ruedas del robot. Algunas de las disposiciones más habituales en robótica móvil son diferencial, triciclo, Ackermann, omnidireccional y orugas. Las configuraciones en triciclo y Ackermann permiten moverse en línea recta con mayor facilidad pero su capacidad para maniobrar es bastante limitada. Las configuraciones omnidireccional y de orugas suponen más complicaciones en los cálculos odométricos. Por ello se ha optado por la configuración diferencial: dos ruedas diametralmente opuestas y ubicadas en un eje perpendicular a la dirección de avance del robot. Cada rueda está dotada de un motor, de forma que los giros se efectúan aplicando diferentes velocidades a cada rueda. En configuración diferencial es complicado mantener la horizontalidad del robot, a no ser que se utilicen técnicas de autobalanceo (mediante giróscopos, como en los *Segway*). Esto se suele solucionar mediante ruedas *castor*, que no están motorizadas y que pueden girar libremente en cualquier dirección. Si se coloca más de una rueda castor, pueden presentarse problemas de tracción en las ruedas motoras, por lo que en el diseño final se ha utilizado una única rueda castor.

Una vez decidida la configuración diferencial del robot, se diseñó la base mediante AutoCAD. El diseño CAD se muestra en la figura A.1a, junto con la distribución de los principales componentes. Las dimensiones de la base se han tratado de minimizar en función de los elementos que iban a utilizarse: ruedas, motores, controladora de motores, batería, sensor láser y cámara, además de otros elementos de menor tamaño como un conversor de 12 a 24 V, *encoders* para medir el desplazamiento del robot, y un mini-ordenador portátil. El aspecto final del robot una vez terminado se muestra en la figura A.1b. Se ha bautizado como *Morlaco* (***Mo**bile **R**obot – **La**ser **Co**ntrolled*), y su nombre se eligió tras realizar las primeras pruebas y observar la sorprendente potencia que le confieren sus motores.



(a) (b)

Figura A.1: a): Diseño CAD de la base del robot. b) Aspecto final del robot.

A.2 SENSORES

Para poder navegar por cualquier tipo de entorno, los robots necesitan sensores que les permitan percibir el medio que los rodea. Los sensores se pueden catalogar como exteroceptivos, que miden estímulos que se originan en el exterior del robot, o como propioceptivos, que sirven para observar el estado interno del robot. En *Morlaco* se incluyen dos sensores exteroceptivos, concretamente un láser de medición de distancias y una cámara de visión monocular, así como un par de *encoders* ópticos incrementales, que son sensores propioceptivos. El aspecto de estos dispositivos se puede observar en la figura A.2, mientras que en el capítulo 4.1 se estudian sus propiedades en profundidad.

(a) (b) (c)

Figura A.2: a) Escáner láser SICK LMS 200. b) Webcam 9000 pro de Logitech. c) Encoder óptico incremental E3 de USdigital.

Una de las dificultades que se ha presentado a la hora de ubicar estos sensores en el robot ha sido cómo sujetarlos firmemente. Por ejemplo, para asegurar el láser se ha utilizado una pieza de acero en forma de T invertida. En el caso de la cámara, la fijación ha sido más complicada debido a sus escasas posibilidades de sujeción. Por ello, se ha optado por colocarla y sujetarla en el interior de una caja de plástico, para impedir cualquier tipo de movimiento o giro. Posteriormente, la cámara se ha fijado a los laterales del láser mediante varías guías de hierro para impedir oscilaciones cuando el robot se encuentra acelerando o frenando.

A.3 ACTUADORES

Para el accionamiento de las ruedas motrices de *Morlaco* hemos elegido motores DC. Elegir unos motores adecuados depende del peso total del robot, así como de la aceleración y velocidad máxima que se desea que alcance. A partir de estos datos, se hace una estimación del par motor que se necesita para mover el robot a la velocidad y aceleración deseadas. Para calcular el par motor, nos basamos en una estimación del peso aproximado del robot. En la tabla A.1a se encuentran recogidos los pesos de los principales elementos del robot. Debido al elevado peso

ELEMENTO	PESO
Laser LMS 200	4.5 kg
Soporte Láser	5 kg
Ruedas	2 kg
Plataforma	5 kg
Batería	15 kg
Motores	5 kg
Otros	0.5 kg
Total	37 kg

(a)

PAR $(N \cdot m)$	RPM	AMPERIOS
0,02	285	6,2
1,37	269	10,2
2,24	261	12,9
3,17	252	15,6
4,29	240	19,5
5,60	227	23,6
6,79	213	27,9

(b)

Tabla A.1: a) Peso de los componentes de *Morlaco*. b) Características del motor NPC-2212.

total, se ha decidido utilizar motores de *National Power Chair* (http://www.npcrobotics.com), que se dedica a la fabricación de motores para sillas de ruedas motorizadas. Estos motores son habitualmente utilizados en la construcción de robots de gran peso en ligas de combate de robots, competiciones muy populares en Estados Unidos y Canadá.

El par motor necesario para mover un robot, $|\tau|$, se puede calcular como:

$$|\tau| = r \cdot F = r \cdot m \cdot a \tag{A.1}$$

donde r es el radio de las ruedas, m la masa total del robot, y a la aceleración. Las ruedas seleccionadas tienen un radio de $0.1\,m$, y como aceleración máxima se ha considerado más que suficiente un valor de $2\,m/s^2$, con lo que el par motor necesario debe valer $|\tau| = 7.4\,N/m$. Como el robot tiene dos motores iguales, el par para cada motor se reduce a la mitad: $|\tau| = 3.7\,N \cdot m$.

Entre los motores disponibles del fabricante, el NPC-2212 parecía el más adecuado debido a sus características de par motor, las revoluciones por minuto que ofrece y el consumo, como se muestra en la tabla A.1b. Este motor es más que suficiente para cumplir nuestros requisitos, ya que podría ofrecernos cerca de 245 RPM. Si convertimos este dato a velocidad máxima se obtendría:

$$v = 2\pi r \cdot rpm/60 = 2\pi \cdot 0.1 \cdot 245/60 \simeq 2.5\,m/s \tag{A.2}$$

Con estos motores *Morlaco* se podría mover a una velocidad máxima de $2.5\,m/s$ con una aceleración de $2\,m/s^2$, todo ello consumiendo aproximadamente unos $17\,A$ por motor. En caso de que la masa total

del robot fuese mayor, el motor todavía tendría par suficiente como para moverlo según nuestros requisitos.

A.3.1 *Sistema de control*

Para el control de los actuadores se ha optado por utilizar una controladora comercial de Roboteq (`http://www.roboteq.com`), concretamente el modelo AX2850, que se muestra en la figura A.3. Dispone de dos

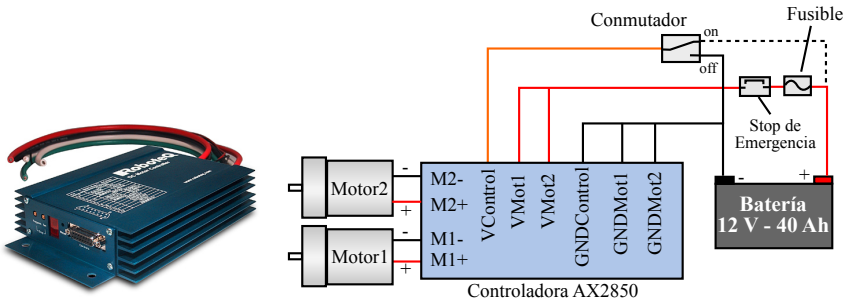


Figura A.3: Controladora de motores AX2850 y esquema eléctrico para conectarla a los motores y la batería del robot.

canales para controlar dos motores con escobillas de manera independiente y cada canal es capaz de ofrecer un máximo de 80 A, con lo cual es más que suficiente para nuestras necesidades. Gracias a la realimentación proporcionada por los *encoders* colocados en los ejes de los motores, la AX2850 es capaz de efectuar un control de lazo cerrado para ajustar la velocidad.

En la parte derecha de la figura A.3 se muestra el esquema eléctrico de conexión de la controladora a los motores y la batería del robot. Cada motor se alimenta de forma independiente de la batería e internamente tiene una etapa de potencia MOSFET individual. Así, las entradas de alimentación (VMot1-2, GNDMot1-2) se convierten en la corriente de salida a los motores (M1+,M1-,M2+,M2-). Por otra parte, la controladora necesita una tensión de alimentación (*Vcontrol*, *GNDControl*) y está conectada a un conmutador, de tal modo que la controladora solamente está apagada cuando *VControl* se conmuta a tierra. Además se puede observar que se ha colocado un stop de emergencia, así como un fusible para proteger la controladora ante posibles daños.

A.4 SOFTWARE DE CONTROL

El objetivo principal de *Morlaco* es efectuar misiones autónomas donde la localización es fundamental para alcanzar los objetivos. Para lograrlo, se han diseñado un conjunto de módulos de bajo y alto nivel, y se han integrado en un RDE desarrollado en GroUsal consistente en la combinación de otros dos entornos robóticos: CARMEN y MISSIONLAB. En el apartado 6.2.2 se analizó brevemente la función de cada módulo desarrollado para el sistema de localización, mientras que en esta sección, en la figura A.4 se presenta la comunicación de cada módulo de bajo nivel con el dispositivo físico que utiliza.

El módulo *camera9000_pro* se encarga de la comunicación por USB con la Logitech Webcam 9000 Pro de *Morlaco*. Básicamente es un driver basado en *video4linux2* que obtiene imágenes en formato YUYV. Sin

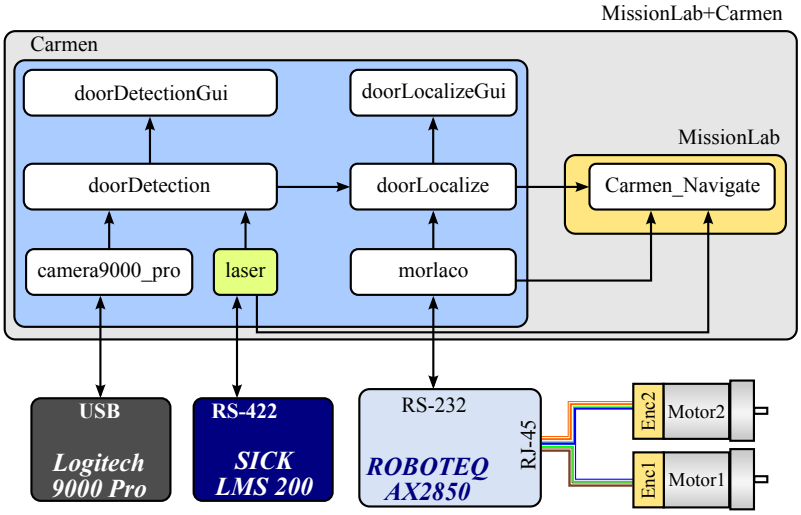


Figura A.4: Comunicación de los módulos de bajo nivel con los dispositivos del robot.

duda este formato es mejor que JPEG, puesto no utiliza compresión, y esto es esencial en el análisis de imágenes para no perder información. Además, el formato YUYV separa el canal de luminancia de los de crominancia, de tal manera que no se necesita convertir de RGB a blanco y negro.

Con respecto al láser SICK LMS 200, durante las primeras fases de desarrollo se implementó un módulo de bajo nivel, pero posteriormente fue sustituido por un módulo existente de CARMEN. La comunicación se realiza por puerto serie, utilizando el estándar RS-422 para obtener la máxima velocidad aceptada por el láser (500KB/s).

Finalmente, el módulo *morlaco* se comunica con la controladora AX2850 e implementa funciones para consultar el desplazamiento relativo de los *encoders* y para enviar comandos de actuación a los motores. La comunicación con la controladora se lleva a cabo por puerto serie, utilizando el estándar RS-232 a 9600 bps, la máxima velocidad aceptada por la AX2850.

APÉNDICE B

ODOMETRÍA

En este apéndice se desarrollan las ecuaciones necesarias para calcular el desplazamiento de nuestro robot en el sistema de referencia global a partir de los desplazamientos relativos de cada rueda en el sistema de referencia del robot.

B.1 DESPLAZAMIENTO RELATIVO

El robot *Morlaco* (ver apéndice A) dispone de sendos *encoders* incrementales en los ejes de sus dos ruedas motrices, colocadas en disposición diferencial. Los *encoders* proporcionan un número de ticks proporcional al número de vueltas que ha dado la rueda. Teniendo en cuenta que los *encoders* son sensores propioceptivos, los datos obtenidos se tienen que interpretar con respecto al sistema de referencia local del robot $\{X_R, Y_R\}$, como se ilustra en la parte izquierda de la figura B.1.

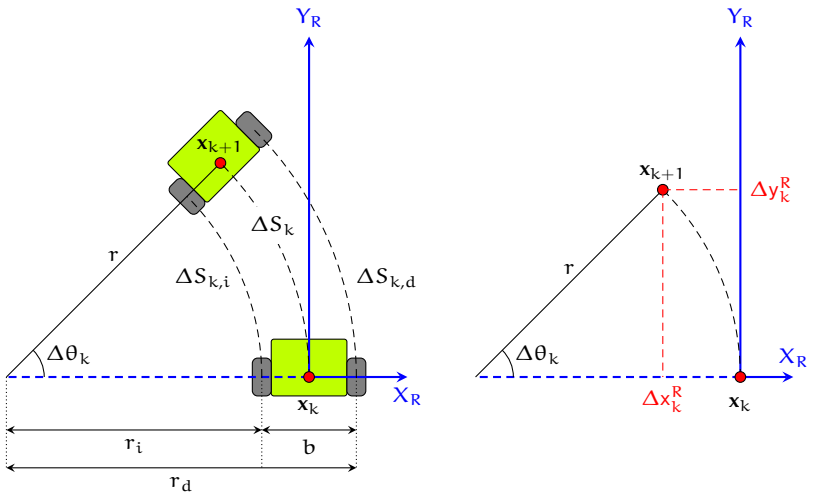


Figura B.1: Desplazamiento relativo del robot con respecto a su sistema de referencia local.

En dicha figura, el robot parte del estado $\mathbf{x}_k$ en el instante de tiempo k y llega al estado $\mathbf{x}_{k+1}$ en el instante de tiempo $k+1$. El intervalo de tiempo $[k, k+1]$ debe ser lo suficientemente pequeño como para poder suponer que la velocidad es constante. En ese intervalo, el robot recorre un arco ΔS_k, correspondiente a un ángulo $\Delta\theta_k$ y radio de giro r, cumpliéndose que:

$$\Delta S_k = r \cdot \Delta\theta_k \tag{B.1}$$

Además se muestran los desplazamientos de la rueda derecha $(\Delta S_{k,d})$ e izquierda $(\Delta S_{k,i})$, la distancia entre ruedas (b), así como el radio de

giro de ambas ruedas (r_i y r_d). Sabiendo que $b = r_d - r_i$, se puede obtener r como:

$$r = r_i + \frac{b}{2} = \frac{r_d + r_i}{2} \tag{B.2}$$

El desplazamiento ΔS_k se calcula en función de las revoluciones de las ruedas del robot, puesto que es donde están colocados los *encoders*. Conociendo la resolución de los *encoders*, las revoluciones se pueden calcular de forma inmediata. Si n es el número de ticks por vuelta de un *encoder*, ΔN_i y ΔN_d el número de ticks que el *encoder* cuenta para las ruedas izquierda y derecha en el intervalo de tiempo $[k, k+1]$, y C el perímetro de las ruedas, entonces la distancia que se desplazan las ruedas izquierda y derecha del robot se calcula como:

$$\begin{aligned} \Delta S_{k,i} &= \frac{\Delta N_i}{n} \cdot C \\ \Delta S_{k,d} &= \frac{\Delta N_d}{n} \cdot C \end{aligned} \tag{B.3}$$

Tomando como base que los valores $\Delta S_{k,i}$ y $\Delta S_{k,d}$ se pueden calcular en cada incremento de tiempo sondeando los *encoders*, es deseable expresar ΔS_k y $\Delta\theta_k$ en función de ambos valores. Sustituyendo (B.2) en (B.1), se obtiene:

$$\Delta S_k = \frac{r_d + r_i}{2}\,\Delta\theta_k = \frac{r_d\,\Delta\theta_k + r_i\,\Delta\theta_k}{2} = \frac{\Delta S_{k,d} + \Delta S_{k,i}}{2} \tag{B.4}$$

Por otra parte, utilizando la distancia entre ruedas (b), el valor de $\Delta\theta_k$ se calcula como:

$$\Delta\theta_k = \Delta\theta_k \cdot \frac{b}{b} = \Delta\theta_k \cdot \frac{r_d - r_i}{b} = \frac{\Delta S_{k,d} - \Delta S_{k,i}}{b} \tag{B.5}$$

Una vez que se dispone de las ecuaciones para calcular el desplazamiento del robot en función de los valores de los *encoders*, se ha de expresar esta información en el sistema de referencia local del robot, tal y como se ilustra en la parte derecha de la figura B.1. Observando dicha figura se deduce que el estado del sistema en el instante de tiempo $k+1$ se puede expresar en el sistema de referencia local del robot como:

$$\mathbf{x}^R_{k+1} = \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta\theta_k \end{bmatrix}^R = \begin{bmatrix} -r(1 - \cos(\Delta\theta_k)) \\ r \cdot \sin(\Delta\theta_k) \\ \Delta\theta_k \end{bmatrix}^R \tag{B.6}$$

B.2 DESPLAZAMIENTO GLOBAL

En la figura B.2 se muestra el mismo desplazamiento del robot mostrado en la figura B.1, pero en el sistema de referencia global y suponiendo que en el instante de tiempo k el robot está orientado en la dirección θ_k.

Observando la figura está claro que en el sistema de referencia global, el estado del sistema en instante de tiempo $k+1$, se puede expresar en función del estado anterior $\mathbf{x}_k$ más un desplazamiento en la posición y la orientación del robot:

$$\mathbf{x}^G_{k+1} = \mathbf{x}^G_k + \begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta\theta_k \end{bmatrix}^G \tag{B.7}$$

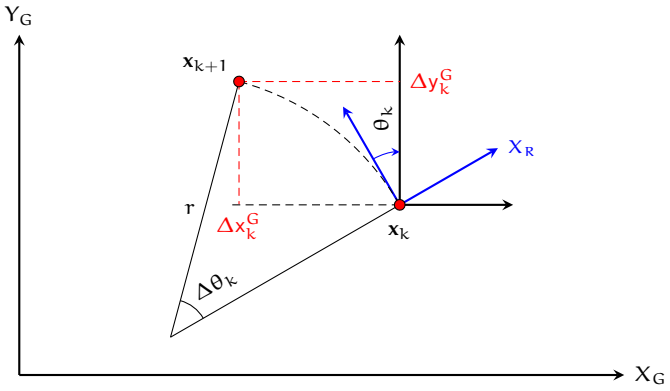


Figura B.2: Desplazamiento relativo del robot con respecto al sistema de referencia global.

En el apartado anterior se calculó el desplazamiento de la posición y la orientación del robot en el sistema del robot $\{X_R, Y_R\}$. Para trasladar estos datos al sistema de referencia $\{X_G, Y_G\}$, basta con girar los ejes $\{X_R, Y_R\}$ un ángulo θ_k en el sentido de las agujas del reloj, alineando así el sistema de referencia local con el global. Esto se logra multiplicando (B.6) por la matriz de rotación correspondiente:

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\\ \Delta\theta_k\end{bmatrix}^G = \begin{bmatrix}\cos(\theta_k) & -\sin(\theta_k) & 0\\ \sin(\theta_k) & \cos(\theta_k) & 0\\ 0 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix}\Delta x_k\\ \Delta y_k\\ \Delta\theta_k\end{bmatrix}^R \tag{B.8}$$

Dado que el valor de $\Delta\theta_k$ permanece constante al cambiar de sistema de referencia, a continuación únicamente se desarrollan Δx_k e Δy_k, con el fin de simplificar las operaciones matriciales:

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\end{bmatrix}^G = \begin{bmatrix}\cos(\theta_k) & -\sin(\theta_k)\\ \sin(\theta_k) & \cos(\theta_k)\end{bmatrix} \cdot \begin{bmatrix}-r(1-\cos(\Delta\theta_k))\\ r\cdot\sin(\Delta\theta_k)\end{bmatrix} =$$

$$\begin{bmatrix}-r\ \cos(\theta_k)+r\ \cos(\theta_k)\ \cos(\Delta\theta_k)-r\ \sin(\theta_k)\ \sin(\Delta\theta_k)\\ -r\ \sin(\theta_k)+r\ \sin(\theta_k)\ \cos(\Delta\theta_k)+r\ \cos(\theta_k)\ \sin(\Delta\theta_k)\end{bmatrix} \tag{B.9}$$

donde utilizando las fórmulas para el coseno de la suma de dos ángulos y el seno de la suma de dos ángulos resulta:

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\end{bmatrix}^G = \begin{bmatrix}-r\ \cos(\theta_k)+r\ \cos(\theta_k+\Delta\theta_k)\\ -r\ \sin(\theta_k)+r\ \sin(\theta_k+\Delta\theta_k)\end{bmatrix} \tag{B.10}$$

A continuación, introduciendo los cambios de variable descritos en (B.11), se obtiene (B.12) y transformando las sumas trigonométricas en productos resulta (B.13).

$$\left.\begin{aligned}\alpha &= \theta_k + \frac{\Delta\theta_k}{2}\\ \beta &= \frac{\Delta\theta_k}{2}\end{aligned}\right\} \Longrightarrow \begin{cases}\theta_k+\Delta\theta_k = \alpha+\beta\\ \theta_k = \alpha-\beta\end{cases} \tag{B.11}$$

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\end{bmatrix}^G = \begin{bmatrix}-r\ \cos(\alpha-\beta)+r\ \cos(\alpha+\beta)\\ -r\ \sin(\alpha-\beta)+r\ \sin(\alpha+\beta)\end{bmatrix} \tag{B.12}$$

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\end{bmatrix}^G = \begin{bmatrix}-2r\ \sin(\alpha)\ \sin(\beta)\\ 2r\ \cos(\alpha)\ \sin(\beta)\end{bmatrix} \tag{B.13}$$

Por otra parte, sabiendo que se cumple $\sin(\theta) \approx \theta$ para ángulos suficientemente pequeños, y haciendo uso de (B.1), se tiene que:

$$2r\ \sin(\beta) = 2r\ \sin\left(\frac{\Delta\theta_k}{2}\right) \overset{\Delta\theta\sim 0}{=} r\,\Delta\theta_k = \Delta S_k \tag{B.14}$$

Simplificando, se obtiene (B.15), y sustituyendo en (B.7) resulta finalmente que el desplazamiento del robot en el marco de referencia global se expresa como (B.16).

$$\begin{bmatrix}\Delta x_k\\ \Delta y_k\end{bmatrix}^G = \begin{bmatrix}-\Delta S_k \cdot \sin\left(\theta_k + \dfrac{\Delta\theta_k}{2}\right)\\ \Delta S_k \cdot \cos\left(\theta_k + \dfrac{\Delta\theta_k}{2}\right)\end{bmatrix} \tag{B.15}$$

$$\mathbf{x}_{k+1}^G = \mathbf{x}_k^G + \begin{bmatrix}-\Delta S_k \cdot \sin\left(\theta_k + \dfrac{\Delta\theta_k}{2}\right)\\ \Delta S_k \cdot \cos\left(\theta_k + \dfrac{\Delta\theta_k}{2}\right)\\ \Delta\theta_k\end{bmatrix} \tag{B.16}$$

Esta ecuación permite calcular el estado del robot en el instante $k+1$ en función del estado en el instante k, y los valores proporcionados por los *encoders*, teniendo en cuenta las ecuaciones B.4 y B.5.

B.3 ERRORES POR APROXIMACIÓN PARA ÁNGULOS PEQUEÑOS

Es muy importante notar que en el modelo del sistema se está utilizando la aproximación para ángulos pequeños $\sin(\theta) \approx \theta$. En la práctica, esto impone unas restricciones en el periodo de muestreo y en la velocidad de giro del robot:

- Si el robot se desplaza en un movimiento perfectamente rectilíneo (cosa prácticamente imposible en el mundo real), los cálculos odométricos no presentarían error alguno, dado que $\sin(0) = 0$.
- Si en el desplazamiento del robot se ha producido algún giro, entonces dado que $\sin(\theta)$ no es exactamente igual a θ, hay que tener en cuenta que las ecuaciones introducen un pequeño error.

APÉNDICE C

DETECCIÓN DE BORDES VERTICALES

En este apéndice se analiza el problema de la detección de bordes en una imagen. Se presentan algunos de los operadores de convolución más habituales utilizados para la detección de bordes en una imagen. Además se describe la convolución separable y se compara su coste computacional con respecto a la convolución estándar. La información aquí presentada se utiliza para comparar tiempos de ejecución en el capítulo 6.1

C.1 INTRODUCCIÓN

Uno de los problemas más estudiados desde los comienzos de la visión artificial es la detección de bordes en imágenes. Tanta dedicación se debe a que una discontinuidad en la intensidad entre dos píxeles contiguos está normalmente relacionada con una discontinuidad física en la escena real. Existe así una conexión entre los bordes detectados y el mundo físico, de tal modo que un borde en la imagen se puede corresponder con objetos a diferentes profundidades, una superficie que cambia repentinamente de orientación, un cambio de material o variaciones en la iluminación de una escena [64]. Así es como la visión humana puede diferenciar el perímetro de un objeto, puesto que cualquier objeto normalmente tiene una intensidad diferente al resto de objetos que lo rodean; de lo contrario sería prácticamente indistinguible.

Un aspecto fundamental en la detección de bordes es precisamente definir en qué consiste un borde. Sin embargo, los datos de una imagen obtenida por una cámara digital son discretos, y para una función definida en un dominio discreto no existe una noción implícita de discontinuidad. Esto significa que no hay una medida inherente para juzgar cuáles son los bordes de una imagen discreta, y por lo tanto el concepto de borde es simplemente *lo que se decida definir que es*. Esta es la principal razón por la que detectar bordes es una tarea no trivial, a menos que los objetos de una escena sean especialmente simples y las condiciones de iluminación estén bien controladas.

En la figura C.1, se muestran dos ejemplos de cómo los bordes de una imagen pueden llegar a ser muy subjetivos. En la imagen superior izquierda aparecen dos torres sobre un terreno montañoso; a su derecha y en la fila del medio, se muestran imágenes con los bordes marcados a mano por 3 personas diferentes. La subjetividad en la detección de bordes es manifiesta: el contorno de las torres y el perfil de la montaña está claro que son bordes, mientras que las ventanas, las aristas de las torres y la vegetación son contornos más ambiguos. Lo mismo ocurre con la imagen del soldado en la fila inferior: las ventanas de la pared, la puerta de la caseta o las franjas en la chaqueta del soldado no son bordes demasiado claros, mientras que el contorno del soldado y de

la caseta sí lo son. Las imágenes están extraídas de la base de datos de segmentación de imágenes de la Universidad de Berkeley (http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/).

Figura C.1: Ejemplo de detección de bordes manual realizada por diferentes personas. Los bordes detectados varían al ser analizados por personas diferentes.

C.2 GRADIENTE DE UNA IMAGEN

Se puede pensar en una imagen como un mapa de alturas, donde la intensidad de un píxel representa la altura. Dado que un borde no es más que un cambio abrupto de intensidad en los píxeles de una imagen, entonces los bordes ocurren en las partes del mapa de alturas donde la pendiente es más pronunciada. La dirección y magnitud de los cambios de pendiente en una imagen I se puede medir mediante el gradiente:

$$\nabla I(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{C.1}$$

En el caso de una imagen, el gradiente mide la dirección de máxima diferencia de intensidades de píxeles, es decir, de negro $I(x,y) = 0$ a blanco $I(x,y) = 255$. El módulo y la dirección del borde se calculan con las expresiones (C.2) y (C.3).

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{C.2}$$

$$\psi = \arg\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{C.3}$$

Al calcular las derivadas se amplifican las altas frecuencias y por lo tanto se amplifica también el ruido, puesto que la proporción de ruido en una señal es mayor en las frecuencias más altas. Por lo tanto, antes de calcular el gradiente suele ser habitual aplicar un filtro pasa-baja, como por ejemplo un filtro gaussiano, con el objetivo de suavizar y reducir el ruido en la imagen.

La derivada parcial de una función continua $F(x, y)$ con respecto a la variable x se define mediante el límite (C.4). Dado que una imagen es una función discreta, no se puede tomar Δx arbitrariamente pequeño, sino que su valor mínimo corresponde a un píxel. Por lo tanto en una imagen I, una aproximación elemental de la derivada horizontal en los píxeles $x = i$ y $y = j$ se puede expresar como (C.5).

$$\frac{\partial F(x, y)}{\partial x} = \lim_{\Delta \to 0} \frac{F(x + \Delta x, y) - F(x, y)}{\Delta x} \tag{C.4}$$

$$\left.\frac{\partial I(x, y)}{\partial x}\right|_{x=i, y=j} \approx I(i + 1, j) - I(i, j) \tag{C.5}$$

El gradiente de una imagen se puede calcular utilizando filtros de convolución para averiguar las derivadas de la imagen en las direcciones x e y, como se mostrará en el siguiente apartado. El tamaño del filtro de convolución afecta a la inmunidad ante el ruido y la precisión del gradiente calculado. Según Kitchen y Malin [57], las máscaras 2×2 producen errores relativamente grandes y que afectan a la dirección del gradiente, mientras que las máscaras 3×3 resultan ser más precisas. Utilizar filtros de convolución de mayor tamaño no es práctico dado que los cálculos son más costosos y no se producen resultados justificablemente mejores [54].

C.3 OPERADORES DE DETECCIÓN DE BORDES

Tal y como se indica en la ecuación (C.5), la derivada horizontal de la imagen en un punto se puede calcular como una diferencia de píxeles. Para ampliar el cálculo a toda la imagen, basta con aplicar el siguiente filtro de convolución a la imagen en cuestión:

$$G_x = \begin{bmatrix} -1 & +1 \end{bmatrix} * I \tag{C.6}$$

donde $*$ es la operación de convolución, $[\,-1\ +1\,]$ es el filtro o kernel de convolución, I es la imagen de entrada, y G_x es la imagen de salida que contiene la derivada horizontal de I. Para calcular la derivada vertical de la imagen, basta con rotar 90^o el kernel de convolución.

Al utilizar el filtro de convolución $[\,-1\ +1\,]$ para calcular la derivada de la imagen, se comete un error de aproximación, tal y como apunta Nixon [84]. El error es menor si en su lugar se utiliza el kernel:

$$k = \begin{bmatrix} -1 & 0 & +1 \end{bmatrix} \tag{C.7}$$

de ahí que se utilice este patrón en los filtros de Prewitt y Sobel, como se verá a continuación.

Además de estos dos filtros de convolución, existen otros muchos para efectuar aproximaciones de primer orden de la derivada de la imagen, y que se pueden utilizar para calcular el gradiente de la imagen. En general, si G_x contiene la derivada horizontal de la imagen, y G_y contiene la derivada vertical, entonces la magnitud y la orientación del gradiente se pueden definir como:

$$|\nabla I(x,y)| = G(x,y) = \sqrt{G_x^2 + G_y^2} \tag{C.8}$$

$$\Theta(x,y) = \arctan\left(\frac{G_y}{G_x}\right) \tag{C.9}$$

Los filtros de convolución que se muestran a continuación son los operadores diferenciales más habitualmente empleados.

C.3.1 *Operador de Roberts*

Es uno de los primeros operadores que aparecieron para la detección de bordes, y consiste en dos filtros de convolución 2×2:

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \tag{C.10}$$

El principal inconveniente del operador de Roberts es que tiene una sensibilidad bastante alta al ruido, porque se utilizan muy pocos píxeles para aproximar el gradiente.

C.3.2 *Operador de Prewitt*

Este operador, junto con el de Sobel, aproxima la primera derivada de la imagen. Utiliza una máscara de convolución de 3×3:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{C.11}$$

C.3.3 *Operador de Sobel*

Es muy similar al operador de Prewitt, pero ligeramente diferente, y normalmente se considera que obtiene mejores resultados. Se define como:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{C.12}$$

C.3.4 *Operador de Laplace*

El operador de Laplace ∇^2 sirve para aproximar la segunda derivada. Normalmente se aplica mediante un filtro de convolución 3×3, que es similar en las direcciones x e y, y que se define como:

$$G_{xy} = \begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix} \tag{C.13}$$

Una desventaja de este operador es que es más sensible al ruido que los operadores basados en el gradiente, y en algunos bordes de la imagen produce bordes dobles.

C.4 CONVOLUCIÓN SEPARABLE

Un filtro de convolución en dos dimensiones es *separable* siempre que se pueda expresar como el producto externo de dos vectores. Un ejemplo es el filtro de Sobel vertical, donde si se eligen $v = [1\ 2\ 1]^T$ y $h = [-1\ 0\ 1]$, se obtiene que:

$$s = v * h = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{C.14}$$

Dado que la convolución es asociativa, se cumple que:

$$I * s = I * (v * h) = (I * v) * h \tag{C.15}$$

lo que significa que el resultado de convolucionar la imagen I con v y a continuación con h es exactamente el mismo que al convolucionar la imagen con el kernel de Sobel directamente.

La ventaja computacional de la convolución separable es evidente: si una imagen tiene N_{px} píxeles y se convoluciona con un kernel de dimensiones $P \times Q$, la convolución normal necesita κ_n operaciones:

$$\kappa_n = N_{px} \times P \times Q \tag{C.16}$$

mientras que con la convolución separable se necesitan solamente κ_s operaciones:

$$\kappa_s = N_{px} \times P + N_{px} \cdot Q = N_{px} \times (P + Q) \tag{C.17}$$

Así, el *speed-up* de la convolución separable frente a la convolución normal solamente depende y aumenta en función de las dimensiones del kernel de convolución, y se define como:

$$\sigma = \frac{\kappa_n}{\kappa_s} = P \cdot Q/(P + Q) \tag{C.18}$$

APÉNDICE D

LA IMAGEN INTEGRAL

Una *tabla de áreas sumadas* (*summed area table*) es un algoritmo para calcular rápida y eficientemente la suma de los valores de un subconjunto de píxeles rectangular y contiguo en una imagen. Fue introducido por *Crow* [32] en el mundo de la computación gráfica en 1984 para su uso en *mipmaps*, una técnica de mapeado de texturas a diferentes resoluciones. La utilidad de estas tablas radica en que cada píxel de una textura de menor resolución se calcula como la media de un área píxeles en la textura de mayor resolución. Esta técnica no fue muy utilizada en visión artificial hasta 20 años después al ser rescatada por *Viola* y *Jones* [109] para la detección de caras, utilizándola conjuntamente con las características pseudo-Haar. Sin embargo, estos autores utilizan la denominación *imagen integral* en lugar de *tabla de áreas sumadas*, para enfatizar su uso en el análisis de imágenes en vez de para el mapeado de texturas, y por ello aquí utilizamos la misma denominación de *Viola-Jones*.

Las letras 'mip' en mipmap son un acrónimo del latín 'multum in parvo', que significa 'muchas cosas en poco espacio'.

D.1 CÁLCULO DE LA IMAGEN INTEGRAL

En una imagen I de dimensiones $w \times h$, la imagen integral es una matriz S de las mismas dimensiones, donde el elemento de coordenadas (x, y) contiene la suma de todos los píxeles que se encuentren por encima y a la izquierda de (x, y) incluyendo dicho píxel, tal y como se muestra en la fig. D.1a. Es decir el valor de la imagen integral en el punto (x, y) almacena la suma:

$$S(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j) \tag{D.1}$$

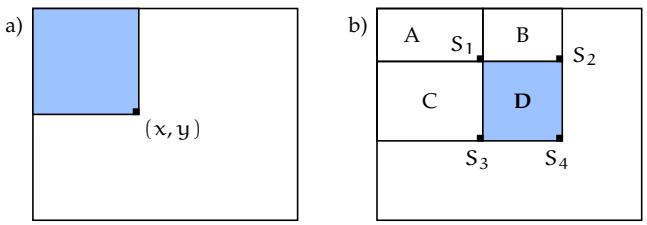


Figura D.1: a) El valor de la imagen integral en el punto (x, y) es la suma de los píxeles contenidos en el rectángulo azul. b) La suma de los píxeles del rectángulo D en función de la imagen integral de los píxeles numerados se calcula como $S_D = S_4 + S_1 - (S_2 + S_3)$.

El coste computacional para calcular la imagen integral varía bastante en función de su formulación. Según Kisačanin [56], utilizando directamente la ecuación (D.1) se necesitarían aproximadamente $\frac{1}{4}w^2 \times h^2$

operaciones. Sin embargo, si se utiliza un enfoque recursivo, se puede mejorar mucho el coste para calcular la imagen integral. Aprovechando las propiedades de la imagen integral, cada uno de sus elementos se puede calcular de una sola pasada recursivamente como:

$$S(x,y) = S(x-1,y) + S(x,y-1) - S(x-1,y-1) + I(x,y) \quad \text{(D.2)}$$

Según esta formulación, se necesitan solamente $3 \times w \times h$ operaciones. No obstante, esta no es la forma más eficaz de calcular la imagen integral, sino que todavía se puede optimizar su cálculo de tal manera que el número de operaciones sea aproximadamente $2 \times w \times h$. Para ello se utilizan el siguiente par de recurrencias, tal y como apuntan Viola-Jones [109]:

$$R(x,y) = R(x,y-1) + I(x,y) \quad \text{(D.3)}$$
$$S(x,y) = S(x-1,y) + R(x,y) \quad \text{(D.4)}$$

donde $R(x,y)$ es la suma acumulada de píxeles de una fila. Utilizando estas fórmulas, el algoritmo D.1 define el pseudocódigo para calcular la imagen integral, suponiendo que las imágenes están almacenadas por filas (como en el lenguaje C). Los parámetros de entrada son la imagen (I), junto con su anchura (w) y altura (h) en píxeles. El algoritmo devuelve como salida la imagen integral (S), de las mismas dimensiones que la imagen de entrada.

Algoritmo D.1 Cálculo de la imagen integral

```
1:  procedure IMAGEN_INTEGRAL(I, w, h, S)
2:      S[0][0] ← I[0][0]
3:      for x ← 1, ··· , w − 1 do
4:          S[x][0] = S[x − 1][0] + I[x][0]
5:      end for
6:      for y ← 1, ··· , h − 1 do
7:          rowsum ← I[y][0]
8:          S[y][0] ← S[y − 1][0] + I[y][0]
9:          for x ← 1, ··· , w − 1 do
10:             rowsum ← rowsum + I[y][x]
11:             S[y][x] ← S[y − 1][x] + rowsum
12:         end for
13:     end for
14: end procedure
```

Al igual que la convolución, la imagen integral es un algoritmo paralelizable. Si es imprescindible trabajar con imágenes muy grandes, o si existe poca potencia computacional en la máquina en la que se ejecute el algoritmo, siempre se puede considerar ejecutar un algoritmo paralelo en una GPU. Por ejemplo, Bilgic *et al.* [13] utilizan la arquitectura de procesamiento paralelo CUDA de Nvidia, y obtienen un *speedup* de casi un orden de magnitud comparado con la versión secuencial.

D.2 SUMAS DE PÍXELES EN ÁREAS RECTANGULARES

Una vez que se ha calculado la imagen integral, cualquier suma de píxeles de un área rectangular de la imagen se puede calcular mediante solamente cuatro accesos al *array* en el que se almacena la imagen integral.

Sea D (ver fig. D.1b) un rectángulo cuyas coordenadas (x_1, y_1) y (x_2, y_2) corresponden a sus vértices superior izquierdo e inferior derecho, respectivamente. Entonces, la suma de sus píxeles se puede calcular en función de los elementos de la imagen integral:

$$S_1 = S(x_1 - 1, y_1 - 1) \tag{D.5}$$
$$S_2 = S(x_2, y_1 - 1) \tag{D.6}$$
$$S_3 = S(x_1 - 1, y2) \tag{D.7}$$
$$S_4 = S(x_2, y_2). \tag{D.8}$$

En S_1 está almacenada la suma correspondiente a los píxeles del rectángulo A, y del mismo modo se pueden establecer las siguientes equivalencias:

$$S_2 = A + B \tag{D.9}$$
$$S_3 = A + C \tag{D.10}$$
$$S_4 = A + B + C + D \tag{D.11}$$

Por lo tanto D se puede expresar como:

$$D = S_4 - (A + B + C) = S_4 + S_1 - (S_2 + S_3) \tag{D.12}$$

y utilizando las coordenadas de los píxeles de la imagen, resulta la ecuación:

$$\begin{aligned} S(x_1, y_1, \ldots, x_2, y_2) \quad = \quad & S(x_2, y_2) + S(x_1 - 1, y_1 - 1) \\ & - \left[S(x_2, y_1 - 1) + S(x_1 - 1, y_2) \right] \end{aligned} \tag{D.13}$$

D.2.1 *Ejemplo práctico*

En la fig. D.2a se muestra una imagen 5×5 donde cada número representa la intensidad de un píxel. Si se suman uno a uno los píxeles dentro del rectángulo amarillo se han de realizar 9 sumas y se obtiene un valor $S = 24$. En la fig. D.2b se ilustra cómo se puede calcular el valor de un elemento de la imagen integral (el recuadrado en azul) mediante la ecuación (D.2). Finalmente, la fig. D.2c muestra la imagen integral calculada por completo, y cómo la suma de los píxeles del rectángulo amarillo de la fig. D.2a se puede calcular mediante 4 sumas, utilizando la ecuación (D.13).

Imagen original

3	2	7	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

a) S = 24

Imagen Integral

3	5	12	14	17
4	11	19	24	31
9	17	28		

b) S=17+19-11+3=28

Imagen Integral

3	5	12	14	17
4	11	19	24	31
9	17	28	38	46
13	24	37	48	62
15	30	44	59	81

c) S=48+3-(14+13)=24

Figura D.2: a) Imagen original. b) Imagen integral: cálculo del píxel central. c) Cálculo de un área utilizando la imagen integral.

D.2.2 *Ahorro computacional*

La imagen integral presenta una ventaja muy importante: permite calcular cualquier suma de píxeles en un área rectangular de una imagen en un tiempo constante, utilizando únicamente cuatro accesos al array en el que está almacenada la imagen integral. No importa si el área es grande o pequeña, puesto que el número de accesos siempre será fijo e igual a 4.

Sin utilizar la imagen integral, el coste para calcular la suma de píxeles de un área con un total de N_{px} píxeles, es exactamente de N_{px} accesos al array que contiene la imagen. Sin embargo, precalculando la imagen integral se necesitan solamente 4 accesos al array de la imagen, de modo que el ahorro en accesos a memoria se puede definir como el porcentaje:

$$\beta = \frac{N_{px} - 4}{N_{px}} * 100 \tag{D.14}$$

En el ejemplo de la figura D.2, $\beta = 55\,\%$, incluso a pesar de que se trata de una imagen muy pequeña. En áreas de dimensiones mayores es obvio que la proporción será aún mayor, dado que el coste para calcular un área utilizando la imagen integral es constante e igual a 4 operaciones. Por ejemplo, para áreas con un número de píxeles $N_{px} \geqslant 40$ el ahorro en operaciones será $\beta \geqslant 90\,\%$.

Como se demuestra en el apartado 6.1.3 mediante resultados experimentales, este ahorro computacional permite buscar líneas verticales en la imagen con una rapidez mucho mayor que la que se puede obtener mediante cualquier otro método.

APÉNDICE E

PROPAGACIÓN DE LA INCERTIDUMBRE

De forma concisa, se puede decir que la propagación de la incertidumbre se refiere al problema de encontrar la distribución estadística de una función de variables aleatorias [5]. Se utiliza para el análisis de sistemas estocásticos en general. En los filtros de Kalman es una parte fundamental para el manejo de la incertidumbre tanto en la fase de predicción como en la de corrección. En el caso de la localización de nuestro robot móvil mediante EKF, la posición del robot tiene asociada una incertidumbre, que hay que propagar cada vez que el robot cambia de posición. Lo mismo ocurre para cada una de las características almacenadas en el SIG: hay que propagar la incertidumbre en la posición del robot al transformar las coordenadas de las características del sistema de referencia global al sistema de referencia del robot.

E.1 PROPAGACIÓN LINEAL

Si el modelo del sistema es lineal, como ocurre en el KF, el caso más simple se da cuando se transforma una única variable aleatoria de entrada, (X), en otra de salida (Y):

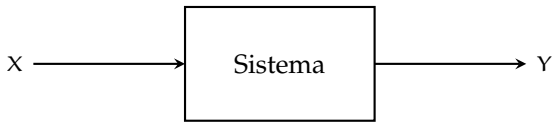


Figura E.1: Sistema con una variable aleatoria de entrada y una de salida.

Suponiendo que se dispone de un modelo matemático del sistema, $Y = f(X)$, y que también se conoce la función de distribución de la entrada $P(X)$, el problema consiste en averiguar la función de distribución de la salida, $P(Y)$.

En los filtros de Kalman, las variables de entrada tienen una distribución normal, que está caracterizada por sus dos primeros momentos estadísticos: la media (μ) y la varianza (σ^2). En la figura E.2a se muestra una variable aleatoria con una distribución normal, indicando su media (μ_x) y su desviación típica (σ_x). Como la función de transformación es lineal (fig. E.2b), la distribución de la variable de salida (fig. E.2c) sigue siendo normal, si bien su media (μ_y) y su varianza (σ_y) son diferentes.

E.2 PROPAGACIÓN NO LINEAL

Cuando el modelo del sistema es no lineal, como ocurre en el EKF, la distribución de la salida del sistema se vuelve muy compleja, especialmente cuando hay más de una variable de entrada. Para resolver este

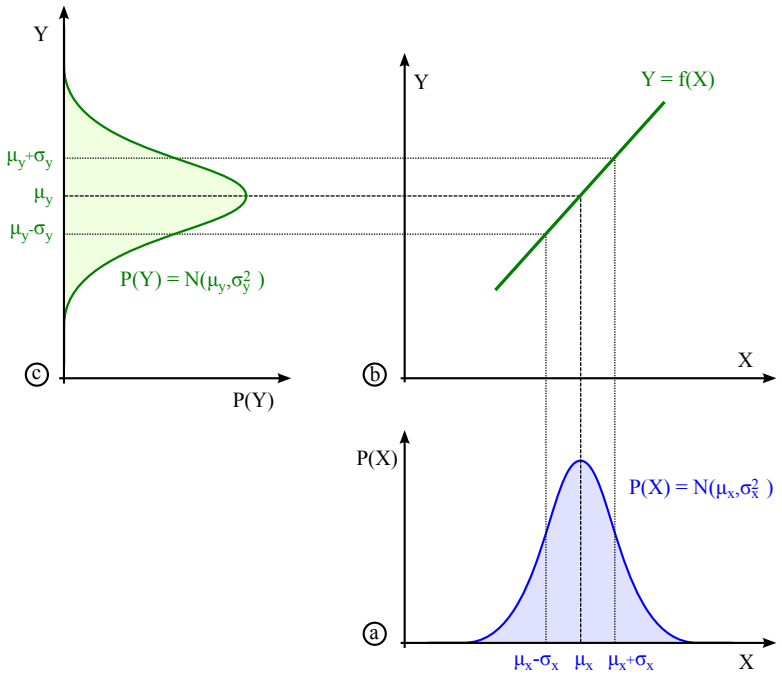


Figura E.2: Propagación del error en una dimensión en el caso de una función lineal. a) Variable aleatoria de entrada. b) Modelo del sistema. c) Variable aleatoria de salida.

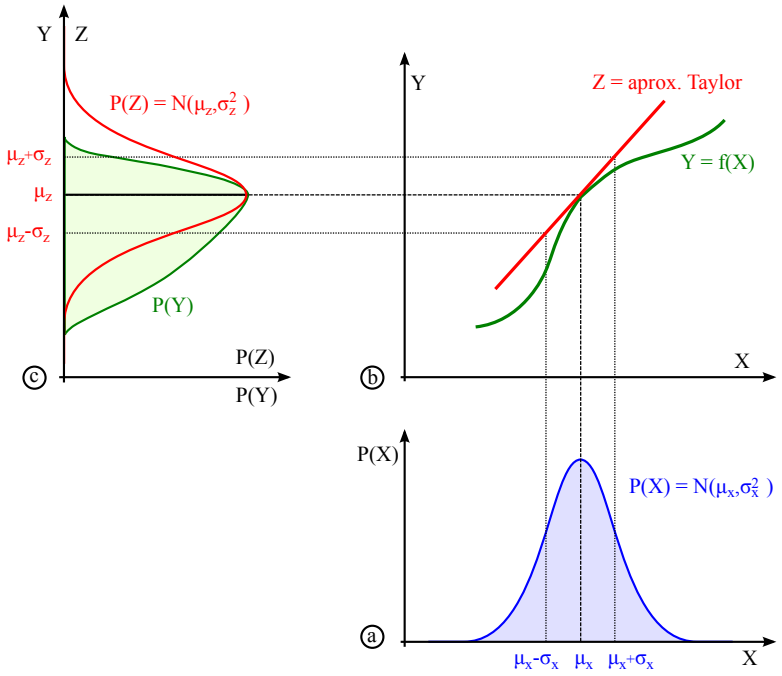


Figura E.3: Propagación del error en una dimensión en el caso de una función no lineal. a) Variable aleatoria de entrada. b) Modelo del sistema (Y) y aproximación de Taylor (Z). c) Variable aleatoria de salida original [P(Y)] y aproximada por Taylor [P(Z)].

problema, se utiliza una aproximación de $f(\cdot)$ mediante una serie de Taylor de primer orden, y se propagan únicamente los dos primeros momentos estadísticos: la media y la varianza. En general, estos dos momentos no describen la distribución de una variable aleatoria, salvo en el caso de que ésta tenga una distribución normal.

En la figura E.3 se muestra un ejemplo en una dimensión de cómo se propaga el error para una variable aleatoria normal cuando el modelo del sistema es no lineal. Se puede observar claramente que cuando se transforma la variable de entrada siguiendo la función no lineal $f(Y)$, la distribución de $P(Y)$ se distorsiona y deja de responder a una distribución normal, siendo además asimétrica. Para conseguir que la variable de salida tenga una distribución normal, hay que aproximar $f(X)$ con una serie de Taylor de primer orden en el punto $X = \mu_x$, resultando:

$$Z \approx f(\mu_x) + \left.\frac{\partial f}{\partial X}\right|_{X=\mu_x} (X - \mu_x) \tag{E.1}$$

Los dos primeros momentos estadísticos de Z, μ_z y σ_z, se calculan como:

$$\mu_z = f(\mu_x) \tag{E.2}$$

$$\sigma_z = \left.\frac{\partial f}{\partial X}\right|_{X=\mu_x} \sigma_x \tag{E.3}$$

A pesar de que la distribución de $P(Z)$ es una aproximación gaussiana de $P(Y)$, normalmente el error cometido al aproximar se encuentra dentro de los límites aceptables dado que la desviación típica de la entrada (μ_x) suele tener valores pequeños.

E.3 LEY DE PROPAGACIÓN DE LA INCERTIDUMBRE

Un sistema estocástico no tiene porqué estar restringido a una sola variable aleatoria de entrada y otra de salida, sino que puede tener n entradas y p salidas:

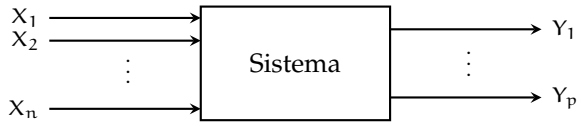


Figura E.4: Sistema con n variables aleatorias de entrada y p variables aleatorias de salida.

Generalizando la aproximación de Taylor de la ecuación E.1 para un sistema multivariable de n entradas y p salidas (consultar [5] para ver el desarrollo completo), se obtiene lo que se conoce como *ley de propagación de la incertidumbre*:

$$C_Y = \nabla f_X \cdot C_X \cdot \nabla f_X^T \tag{E.4}$$

donde C_X (E.5) tiene dimensiones $n \times n$ y es la matriz de covarianza correspondiente a las n variables aleatorias de entrada; C_Y (E.6) tiene dimensiones $p \times p$ y es la matriz de covarianza de las p variables aleatorias de salida; ∇f_X es el Jacobiano (E.7) de la función que modela el sistema y tiene dimensiones $p \times n$.

La interpretación informal de la ecuación (E.4) es que la incertidumbre de las variables de entrada se propaga a través del sistema mediante una aproximación de Taylor multidimensional (por medio de los Jacobianos) y se obtiene la incertidumbre de las variables de salida. Un aspecto muy importante de la ley de propagación de la incertidumbre es que al utilizar una aproximación de primer orden, se garantiza que las variables de salida sean gaussianas siempre que las variables de entrada también lo sean, como ocurre en el EKF.

$$C_X = \begin{bmatrix} \sigma_{x_1}^2 & \sigma_{x_1x_2} & \cdots & \sigma_{x_1x_n} \\ \sigma_{x_2x_1} & \sigma_{x_2}^2 & \cdots & \sigma_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{x_nx_1} & \sigma_{x_nx_2} & \cdots & \sigma_{x_n}^2 \end{bmatrix} \tag{E.5}$$

$$C_Y = \begin{bmatrix} \sigma_{y_1}^2 & \sigma_{y_1y_2} & \cdots & \sigma_{y_1y_p} \\ \sigma_{y_2y_1} & \sigma_{y_2}^2 & \cdots & \sigma_{y_2y_p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{y_py_1} & \sigma_{y_py_2} & \cdots & \sigma_{y_p}^2 \end{bmatrix} \tag{E.6}$$

$$\nabla f_X = \begin{bmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \cdots & \frac{\partial f_1}{\partial X_n} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \cdots & \frac{\partial f_2}{\partial X_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_p}{\partial X_1} & \frac{\partial f_p}{\partial X_2} & \cdots & \frac{\partial f_p}{\partial X_n} \end{bmatrix} \tag{E.7}$$

APÉNDICE F

VISUALIZACIÓN DE LA INCERTIDUMBRE

El objetivo de este apéndice es describir cómo visualizar la incertidumbre en la posición del robot. Para ello, se desarrollan las ecuaciones que permiten calcular una *elipse de error* que garantice que nuestro robot se encuentra en el interior de dicha elipse con la probabilidad deseada.

F.1 INTRODUCCIÓN

Un aspecto clave de los filtros de Kalman, es comprender que las variables que intervienen tanto en el modelo del sistema como en el modelo de observación son variables aleatorias con una distribución normal multivariante: $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P})$ (ver apartado 3.1.2). En el caso multidimensional general, la función de densidad de probabilidad se define como:

$$f_{\mathbf{x}}(x_1, \cdots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{P}|}} e^{-\frac{1}{2}[(\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{P}^{-1} (\mathbf{x}-\hat{\mathbf{x}})]} \tag{F.1}$$

donde n es el número de dimensiones, $\hat{\mathbf{x}}$ el vector media, $\mathbf{P}$ la matriz de covarianza, y $|\mathbf{P}|$ el determinante de $\mathbf{P}$. En este espacio n-dimensional, si se conectan los puntos con el mismo valor de probabilidad, se obtienen elipsoides $n - dimensionales$ centrados en $\hat{\mathbf{x}}$ y cuya forma está definida por $\mathbf{P}$.

F.2 INCERTIDUMBRE EN LA POSICIÓN DEL ROBOT

A lo largo de esta tesis, para indicar en el EKF la posición y orientación del robot en un sistema de referencia global cartesiano, se ha elegido una variable de estado tridimensional:

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{F.2}$$

Esta variable es una variable aleatoria gaussiana que contiene simultáneamente la posición del robot y la incertidumbre. No obstante,

aprovechando la propiedad (F.3) de las distribuciones normales, se pueden tratar por separado la posición (F.4) y la incertidumbre (F.5).

$$\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \mathbf{P}) = \mathcal{N}(\hat{\mathbf{x}}, 0) + \mathcal{N}(0, \mathbf{P}) \tag{F.3}$$

$$\hat{\mathbf{x}} = \mathrm{E}[\mathbf{x}] = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{bmatrix} \sim \mathcal{N}(\hat{\mathbf{x}}, 0) \tag{F.4}$$

$$\mathbf{P} = \mathrm{E}\left[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T\right] = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} \sim \mathcal{N}(0, \mathbf{P}) \tag{F.5}$$

Para representar gráficamente la incertidumbre recogida en la matriz de covarianza $\mathbf{P}$, se utiliza la submatriz $\mathbf{C}$, construida a partir de los términos en x e y:

$$\mathbf{C} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} \sim \mathcal{N}(0, \mathbf{C}) \tag{F.6}$$
$$\sigma_{xy} = \rho \sigma_x \sigma_y$$

En este caso la matriz de covarianza $\mathbf{C}$ corresponde a una distribución normal bivariante, con media $(0, 0)$ y covarianza $\mathbf{C}$, cuya función de densidad de probabilidad es:

$$f(x, y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right]} \tag{F.7}$$

donde $\rho = \sigma_{xy}/(\sigma_x\sigma_y)$ es el coeficiente de correlación entre x e y. La representación gráfica tridimensional de una bivariante gaussiana se muestra en la figura F.1. Debido al alto coste computacional, a la hora de visualizar en tiempo real la incertidumbre del robot, no se pueden mostrar gráficos en 3D, y por ello normalmente se muestran *elipses de error*, que corresponden a las isolíneas de probabilidad constante que se pueden observar en la figura F.2. En el siguiente apartado se demuestra cómo calcular los parámetros de la elipse de error correspondiente a una isolínea con un determinado porcentaje de probabilidad.

F.3 CÁLCULO DE LA ELIPSE DE ERROR

En la ecuación que describe la función de densidad de probabilidad para una distribución multivariante normal (F.1), el término en el exponente responde a la ecuación de una *forma cuadrática*:

$$(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{P}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \in \chi^2(n) \tag{F.8}$$

que tiene una distribución χ^2 con n grados de libertad [48], siendo n la dimensión de $\mathbf{x}$ y el rango de $\mathbf{P}$.

En el caso de una distribución normal bivariante con media $(0, 0)$ (ecuación F.7), se obtiene una forma cuadrática en dos variables y con una distribución χ^2 con 2 grados de libertad:

$$\frac{1}{(1-\rho^2)}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right] \in \chi^2(2) \tag{F.9}$$

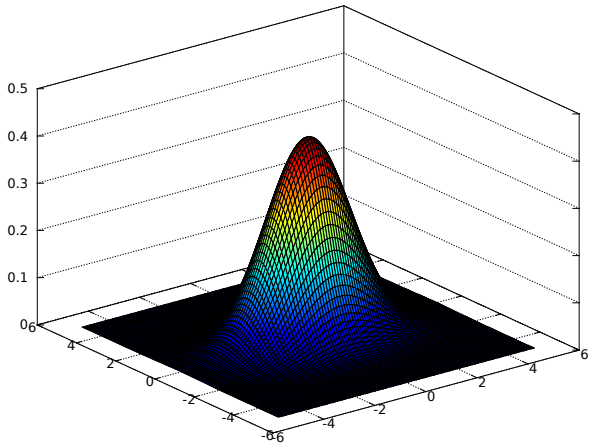


Figura F.1: Ejemplo de representación tridimensional de la función de densidad de probabilidad de una variable aleatoria normal en dos dimensiones con media $(0, 0)$.

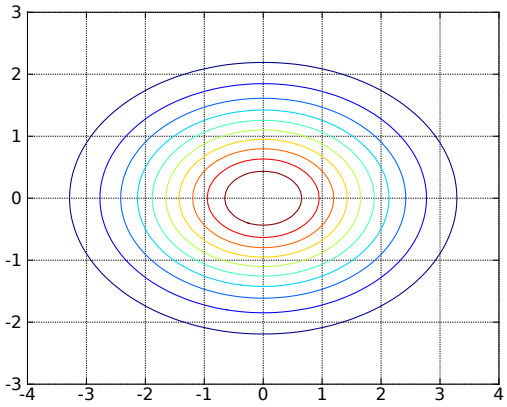


Figura F.2: Isolíneas de probabilidad constante correspondientes a la fig. F.1

Esta ecuación es genérica y no especifica ninguna elipse de error en concreto. Para particularizar esta forma cuadrática para una elipse con un determinado valor de probabilidad P, se necesita determinar el valor de k^2 en la función de distribución acumulada para la variable aleatoria χ^2 con dos grados de libertad, cuya fórmula es:

$$P = 1 - e^{\frac{-k^2}{2}} \tag{F.10}$$

donde tomando logaritmos, se puede expresar k^2 como:

$$\boxed{k^2 = 2\ln(1 - P)} \tag{F.11}$$

Así, una elipse de error con probabilidad P se calcula como:

$$\frac{1}{(1-\rho^2)}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x \sigma_y}\right] = k^2 \tag{F.12}$$

y desarrollando esta última ecuación hasta obtener un denominador común, se obtiene:

$$\frac{x^2\sigma_y^2 - 2xy\rho\sigma_x\sigma_y + y^2\sigma_x^2}{\sigma_x^2\sigma_y^2 - \rho^2\sigma_x^2\sigma_y^2} = k^2 \tag{F.13}$$

que a su vez se corresponde con la familia de elipses:

$$Ax^2 + 2Bxy + Cy^2 - k^2 = 0 \tag{F.14}$$

con coeficientes:

$$\boxed{\begin{aligned} A &= \frac{1}{(1-\rho^2)\sigma_x^2} \\ B &= -\frac{\rho}{(1-\rho^2)\sigma_x\sigma_y} \\ C &= \frac{1}{(1-\rho^2)\sigma_y^2} \end{aligned}} \tag{F.15}$$

Los parámetros A, B y C permiten relacionar los valores de la matriz de covarianza (F.6) con la ecuación general de una elipse. Sin embargo, con esta información no se sabe aún la forma de la elipse, y por ello a continuación se calcularán los ejes x e y de la elipse y su rotación θ con respecto al sistema de referencia.

F.3.1 *Ángulo de rotación de la elipse*

En el sistema de referencia $\{U, V\}$, la ecuación de una elipse cuyos ejes coinciden con los ejes U y V es (ver fig. F.3):

$$\left(\frac{u}{a}\right)^2 + \left(\frac{v}{b}\right)^2 = 1 \tag{F.16}$$

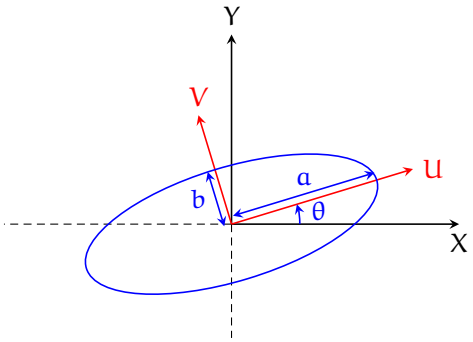


Figura F.3: Parámetros de una elipse rotada: eje x (a), eje y (b) y ángulo de rotación (θ).

Con respecto al sistema de referencia $\{X, Y\}$, los ejes $\{U, V\}$ forman un cierto ángulo θ:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{F.17}$$

Sustituyendo en (F.16):

$$\left(\frac{x\cos(\theta) + y\sin(\theta)}{a}\right)^2 + \left(\frac{y\cos(\theta) - x\sin(\theta)}{b}\right)^2 = 1 \tag{F.18}$$

Desarrollando:

$$\begin{aligned} &b^2\left[x^2\cos^2(\theta) + y^2\sin^2(\theta) + 2xy\cos(\theta)\sin(\theta)\right] \\ &+ a^2\left[y^2\cos^2(\theta) + x^2\sin^2(\theta) - 2xy\cos(\theta)\sin(\theta)\right] = a^2b^2 \end{aligned} \tag{F.19}$$

$$b^2x^2\cos^2(\theta) + b^2y^2\sin^2(\theta) + 2b^2xy\cos(\theta)\sin(\theta) + a^2y^2\cos^2(\theta) + a^2x^2\sin^2(\theta) - 2a^2xy\cos(\theta)\sin(\theta) = a^2b^2 \tag{F.20}$$

$$x^2\left[a^2\sin^2(\theta) + b^2\cos^2(\theta)\right] + 2xy\cos(\theta)\sin(\theta)(b^2 - a^2) + y^2\left[a^2\cos^2(\theta) + b^2\sin^2(\theta)\right] = a^2b^2 \tag{F.21}$$

Esta ecuación es similar a la ecuación (F.14), donde el valor de los coeficientes A, B, C y k^2 es:

$$\boxed{\begin{aligned} A &= a^2\sin^2(\theta) + b^2\cos^2(\theta) \\ B &= \cos(\theta)\sin(\theta)(b^2 - a^2) \\ C &= a^2\cos^2(\theta) + b^2\sin^2(\theta) \\ k^2 &= a^2b^2 \end{aligned}} \tag{F.22}$$

De la ecuación (F.22) se pueden obtener las siguientes igualdades:

$$\begin{aligned} A + C &= a^2\left[\sin^2(\theta) + \cos^2(\theta)\right] + b^2[\sin^2(\theta) + \cos^2(\theta)\right] \\ &= a^2 + b^2 \end{aligned} \tag{F.23}$$

$$\begin{aligned} A - C &= a^2\left[\sin^2(\theta) - \cos^2(\theta)\right] + b^2[\cos^2(\theta) - \sin^2(\theta)\right] \\ &= -a^2\left[\cos^2(\theta) - \sin^2(\theta)\right] + b^2[\cos^2(\theta) - \sin^2(\theta)\right] \\ &= \left[\cos^2(\theta) - \sin^2(\theta)\right](b^2 - a^2) \\ &= \cos(2\theta)(b^2 - a^2) \end{aligned} \tag{F.24}$$

$$B = \frac{\sin(2\theta)}{2}(b^2 - a^2) \tag{F.25}$$

Utilizando (F.24) y (F.25):

$$\begin{aligned} \frac{A - C}{\cos(2\theta)} &= \frac{2B}{\sin(2\theta)} \\ \tan 2\theta &= \frac{2B}{A - C} \end{aligned} \tag{F.26}$$

De donde finalmente se obtiene el ángulo de rotación de la elipse, θ, como:

$$\boxed{\theta = \frac{1}{2}\arctan\left(\frac{2B}{A - C}\right)} \tag{F.27}$$

F.3.2 *Eje x de la elipse*

El eje x de la elipse se representa mediante la variable a en la ecuación (F.18). Expresando las ecuaciones (F.24) y (F.25) como:

$$\left.\begin{aligned} 2B &= \sin(2\theta)(b^2 - a^2) \\ A - C &= \cos(2\theta)(b^2 - a^2) \end{aligned}\right\} \tag{F.28}$$

y elevándolas al cuadrado y sumándolas:

$$(A - C)^2 + (2B)^2 = (b^2 - a^2)^2\left[\cos^2(2\theta) + \sin^2(2\theta)\right] \tag{F.29}$$

$$A^2 + C^2 - 2AC + 4B^2 = (b^2 - a^2)^2 \tag{F.30}$$

Eligiendo T como:

$$\boxed{T = \sqrt{A^2 + C^2 - 2AC + 4B^2}} \tag{F.31}$$

Se tiene que:

$$T^2 = (b^2 - a^2)^2 \tag{F.32}$$

$$T = b^2 - a^2 \tag{F.33}$$

Por otra parte, de (F.22) se obtiene $b^2 = k^2/a^2$, y sustituyendo en (F.23) y (F.33):

$$A + C = a^2 + \frac{k^2}{a^2} \tag{F.34}$$

$$T = \frac{k^2}{a^2} - a^2 \tag{F.35}$$

Sumando ambas ecuaciones y despejando se obtiene finalmente el eje x de la elipse como:

$$\boxed{a^2 = \frac{2k^2}{A + C + T}} \tag{F.36}$$

F.3.3 *Eje y de la elipse*

El eje y de la elipse se representa mediante la variable b en la ecuación (F.18). De (F.22) se obtiene $a^2 = k^2/b^2$, y sustituyendo en (F.23) y (F.33):

$$A + C = \frac{k^2}{b^2} + b^2 \tag{F.37}$$

$$T = b^2 - \frac{k^2}{b^2} \tag{F.38}$$

Restando ambas ecuaciones y despejando:

$$\boxed{b^2 = \frac{2k^2}{A + C - T}} \tag{F.39}$$

F.4 RESUMEN

Para una matriz de covarianza bidimensional C:

$$\mathbf{C} = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix} \tag{F.40}$$

Se calculan los coeficientes A, B C y T como:

$$A = \frac{1}{(1-\rho^2)\sigma_x^2} \tag{F.41}$$

$$B = -\frac{\rho}{(1-\rho^2)\sigma_x\sigma_y} \tag{F.42}$$

$$C = \frac{1}{(1-\rho^2)\sigma_y^2} \tag{F.43}$$

$$T = \sqrt{A^2 + C^2 - 2AC + 4B^2} \tag{F.44}$$

A continuación, se calcula k^2 en función del valor de probabilidad $P \in [0, 1]$ que hace que la elipse de error capture un determinado porcentaje de incertidumbre:

$$k^2 = 2\ln(1 - P) \tag{F.45}$$

Finalmente el ángulo de rotación, y los ejes x e y de la elipse se calculan mediante las expresiones:

$$\theta = \frac{1}{2}\arctan\left(\frac{2B}{A - C}\right) \tag{F.46}$$

$$a^2 = \frac{2k^2}{A + C + T} \tag{F.47}$$

$$b^2 = \frac{2k^2}{A + C - T} \tag{F.48}$$

BIBLIOGRAFÍA

[1] "OpenGIS simple feature specification for sql revision 1.1 (opengis project document 99-049)". Informe técnico, Open GIS Consortium, Inc., Wayland, MA, 1999. (Citado en las páginas 27, 31 y 117.)

[2] "Environmental Systems Research Institute (ESRI) Dictionary of GIS Terminology". http://support.esri.com/en/knowledgebase/GISDictionary/term/GIS, Mayo 2011. (Citado en la página 23.)

[3] "Velodyne - high definition lidar". http://velodynelidar.com, Junio 2012. (Citado en la página 62.)

[4] Anguelov, D.; Koller, D.; Parker, E. y Thrun, S. "Detecting and modeling doors with mobile robots". En: *Proceedings of the IEEE International Conference on Robotics and Automation* (2004), vol. 4, pp. 3777–3784. (Citado en la página 13.)

[5] Arras, K. O. "An introduction to error propagation: Derivation, meaning and examples of cy = fx cx fx"'. Informe técnico, Autonomous Systems Lab, Institute of Robotic Systems, Swiss Federal Institute of Technology Lausanne (EPFL), 1998. (Citado en las páginas 185 y 187.)

[6] Arras, K. O. y Tomatis, N. "Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision". En: *Proceedings of the 1999 Third European Workshop on Advanced Mobile Robots* (1999), pp. 177–185. (Citado en la página 15.)

[7] Arras, K. O.; Tomatis, N.; Jensen, B. y Siegwart, R. "Multisensor on-the-fly localization: Precision and reliability for applications". *Robotics and Autonomous Systems 34*, 2-3 (Feb. 2001), 131–143. (Citado en las páginas 5 y 15.)

[8] Atiya, S. y Hager, G. D. "Real-time vision-based robot localization". *IEEE Transactions on Robotics and Automation 9*, 6 (Dic. 1993), 785–800. (Citado en la página 15.)

[9] Bailey, T. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Tesis Doctoral, Australian Centre for Field Robotics, Department of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, Ago. 2002. (Citado en las páginas 9, 10, 128 y 129.)

[10] Bailey, T. y Durrant-Whyte, H. F. "Simultaneous localization and mapping (SLAM): part II". *Robotics Automation Magazine, IEEE 13*, 3 (Sept. 2006), 108–117. (Citado en la página 4.)

[11] Bar-Shalom, Y.; Li, X.-R. y Kirubarajan, T. *Estimation with Applications To Tracking and Navigation*. John Wiley & Sons, 2001. (Citado en las páginas 37, 41, 43, 44 y 45.)

[12] Barber, R.; Mata, M.; Boada, M.; Armingol, J. y Salichs, M. "A perception system based on laser information for mobile robot topologic navigation". En: *Proceedings of the IEEE 28th Annual Conference of the Industrial Electronics Society* (Nov. 2002), vol. 4, pp. 2779–2784. (Citado en la página 12.)

[13] Bilgic, B.; Horn, B. K. y Masaki, I. "Efficient integral image computation on the gpu". En: *Proceedings of the 2010 IEEE Intelligent Vehicles Symposium* (Junio 2010), pp. 528–533. (Citado en la página 182.)

[14] Bonin-Font, F.; Ortiz, A. y Oliver, G. "Visual navigation for mobile robots: A survey". *Journal of Intelligent & Robotic Systems 53* (2008), 263–296. (Citado en la página 15.)

[15] Bonnifait, P.; Jabbour, M. y Cherfaoui, V. "Autonomous navigation in urban areas using GIS-managed information". *International Journal of Vehicle Autonomous Systems 6*, 1-2 (2008), 83–103. (Citado en la página 11.)

[16] Borenstein, J.; Everett, H. y Feng, L. "Where am I? Sensors and methods for mobile robot positioning". Informe técnico, University of Michigan, 1996. (Citado en las páginas 10 y 51.)

[17] Borges, G. A. y Aldon, M.-J. "Line extraction in 2d range images for mobile robotics". *Journal of Intelligent and Robotic Systems 40*, 3 (Julio 2004), 267–297. (Citado en la página 100.)

[18] Borkowski, A.; Siemiatkowska, B. y Szklarski, J. "Towards semantic navigation in mobile robotics". En *Graph Transformations and Model-Driven Engineering*, G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, Eds., vol. 5765 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 719–748. (Citado en la página 93.)

[19] Bringhurst, R. *The Elements of Typographic Style*. Version 3.2. Hartley & Marks Publishers, Point Roberts, WA, USA, 2008. (Citado en la página 207.)

[20] Burgard, W.; Cremers, A. B.; Fox, D.; Hähnel, D.; Lakemeyer, G.; Schulz, D.; Steiner, W. y Thrun, S. "Experiences with an interactive museum tour-guide robot". *Artificial Intelligence 114*, 1–2 (1999), 3–55. (Citado en la página 4.)

[21] Callaghan, V.; Clarke, G. y Chin, J. "Some socio-technical aspects of intelligent buildings and pervasive computing research". *Intelligent Buildings International 1*, 1 (2009), 56–74. (Citado en la página 3.)

[22] Campbell, J. K.; Synnott, S. P. y Bierman, G. J. "Voyager orbit determination at jupiter". *Automatic Control, IEEE Transactions on 28*, 3 (Mar. 1983), 256–268. (Citado en la página 43.)

[23] CAPPELLE, C.; EL BADAOUI EL NAJJAR, M.; POMORSKI, D. Y CHARPILLET, F. "Intelligent geolocalization in urban areas using global positioning systems, three-dimensional geographic information systems, and vision". *Journal of Intelligent Transportation Systems 14*, 1 (2010), 3–12. (Citado en la página 11.)

[24] CARIÑENA, P.; REGUEIRO, C. V.; OTERO, A.; BUGARÍN, A. J. Y BARRO, S. "Landmark detection in mobile robotics using fuzzy temporal rules". *IEEE Transactions on Fuzzy Systems 12*, 4 (Ago. 2004), 423–435. (Citado en la página 12.)

[25] CASTELLANOS, J. A.; NEIRA, J.; STRAUSS, O. Y TARDÓS, J. D. "Detecting high level features for mobile robot localization". En: *IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1996* (Dic. 1996), pp. 611–618. (Citado en las páginas 10 y 15.)

[26] CASTELLANOS, J. A. Y TARDÓS, J. D. *Mobile Robot Localization and Map Building – A Multisensor Fusion Approach*. Kluwer Academic Publishers, 2000. (Citado en la página 64.)

[27] CHEN, Z. Y BIRCHFIELD, S. T. "Visual detection of lintel-occluded doors from a single image". En: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2008), pp. 1–8. (Citado en la página 14.)

[28] CHEN, Z.; LI, Y. Y BIRCHFIELD, S. T. "Visual detection of lintel-occluded doors by integrating multiple cues using a data-driven markov chain monte carlo process". *Robotics and Autonomous Systems 59*, 11 (2011), 966–976. (Citado en la página 14.)

[29] CHOSET, H.; LYNCH, K. M.; HUTCHINSON, S.; KANTOR, G.; BURGARD, W.; KAVRAKI, L. E. Y THRUN, S. *Principles of Robot Motion - Theory, Algorithms and Implementations*. MIT Press, 2005. (Citado en la página 40.)

[30] COOK, D. J.; AUGUSTO, J. C. Y JAKKULA, V. R. "Ambient intelligence: Technologies, applications, and opportunities". *Pervasive and Mobile Computing 5*, 4 (2009), 277–298. (Citado en la página 3.)

[31] COOPER, A. J. *A Comparison of Data Association Techniques for Simultaneous Localization and Mapping*. Tesis Doctoral, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, 2005. (Citado en la página 10.)

[32] CROW, F. C. "Summed-area tables for texture mapping". *SIGGRAPH Comput. Graph. 18* (Enero. 1984), 207–212. (Citado en la página 181.)

[33] CROWLEY, J. L. Y CHENAVIER, F. "Position estimation for a mobile robot using vision and odometry". En: *Proceedings of the 1992 IEEE International Conference on Robotics and Automation* (1992), vol. 3, pp. 2588–2593. (Citado en la página 15.)

[34] DESOUZA, G. N. Y KAK, A. C. "Vision for mobile robot navigation: a survey". *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 2 (feb 2002), 237–267. (Citado en la página 15.)

[35] Drake, K. C.; McVey, E. S. y Iñigo, R. M. "Sensing error for a mobile robot using line navigation". *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7*, 4 (Julio 1985), 485–490. (Citado en la página 9.)

[36] Duda, R. O. y Hart, P. E. "Use of the hough transformation to detect lines and curves in pictures". *Communications of the ACM 15*, 1 (Enero. 1972), 11–15. (Citado en la página 99.)

[37] Endo, Y.; Ulam, P. D.; Arkin, R. C.; Balch, T. R. y Powers, M. D. "An empirical evaluation of context-sensitive pose estimators in an urban outdoor environment". Informe Técnico GIT-GVU-05-13, Georgia Institute of Technology, 2005. (Citado en la página 148.)

[38] Everett, H. *Sensors for mobile robots - Theory and application*. A K Peters, Ltd., 1995. (Citado en las páginas 61 y 88.)

[39] Fernández-Caramés, C.; Moreno, V.; Curto, B. y Vicente, J. A. "Clustering and line detection in laser range measurements". *Robotics and Autonomous Systems 58*, 5 (2010), 720–726. (Citado en la página 100.)

[40] Filliat, D. y Meyer, J.-A. "Map-based navigation in mobile robots: I. A review of localization strategies". *Cognitive Systems Research*, 4 (2003), 243–282. (Citado en las páginas 4 y 15.)

[41] Fischler, M. A. y Bolles, R. C. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM 24*, 6 (Junio 1981), 381–395. (Citado en la página 99.)

[42] Fukui, I. "Tv image processing to determine the position of a robot vehicle". *Pattern Recognition 14*, 1-6 (1981), 101–109. (Citado en la página 10.)

[43] Galati, S. R. *Geographic Information Systems Demystified*. Artech House, 2006. (Citado en la página 21.)

[44] Gelb, A. *Applied Optimal Estimation*. The MIT Press, 1974. (Citado en la página 41.)

[45] Gowdy, J. "IPT: An object oriented toolkit for interprocess communications". Informe Técnico CMU-RI-TR-96-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Mar. 1996. (Citado en la página 148.)

[46] Grewal, M. S. y Andrews, A. P. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, 2001. (Citado en la página 37.)

[47] Guan, C.-H.; Gong, J.-W.; Chen, Y.-D. y Chen, H.-Y. "An application of data fusion combining laser scanner and vision in real-time driving environment recognition system". En: *Proc. of the 8th International Conference on Machine Learning and Cybernetics* (Julio 2009), vol. 6, pp. 3116–3121. (Citado en la página 73.)

[48] Gut, A. *An Intermediate Course in Probability*. Springer-Verlag, 1995. (Citado en la página 190.)

[49] Güting, R. H. "An introduction to spatial database systems". *VLDB Journal* 3, 4 (Oct. 1994), 357–399. (Citado en la página 28.)

[50] Gutmann, J.-S.; Burgard, W.; Fox, D. y Konolige, K. "An experimental comparison of localization methods". En: *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 1998), vol. 2, pp. 736–743. (Citado en la página 15.)

[51] Hensler, J.; Blaich, M. y Bittel, O. "Real-time door detection based on adaboost learning algorithm". En *International Conference in Research and Education in Robotics - EUROBOT 2009*, vol. 82 of *Communications in Computer and Information Science*. Springer, 2010, pp. 61–73. (Citado en la página 14.)

[52] Julier, S. J. y Uhlmann, J. K. "Unscented filtering and nonlinear estimation". *Proceedings of the IEEE 92*, 3 (Mar. 2004), 401–422. (Citado en la página 4.)

[53] Kabuka, M. R. y Arenas, Á. E. "Position verification of a mobile robot using standard pattern". *Robotics and Automation, IEEE Journal of* 3, 6 (Dic. 1987), 505–516. (Citado en la página 10.)

[54] Kahn, P.; Kitchen, L. y Riseman, E. M. "Real-time feature extraction: a fast line finder for vision-guided robot navigation". Informe técnico, University of Massachusetts, Amherst, MA, USA, Julio 1987. (Citado en la página 177.)

[55] Kim, D. y Nevatia, R. "Recognition and localization of generic objects for indoor navigation using functionality". *Image and Vision Computing 16*, 11 (Ago. 1998), 729–743. (Citado en la página 13.)

[56] Kisačanin, B. "Integral image optimizations for embedded vision applications". En: *IEEE Southwest Symposium on Image Analysis and Interpretation 2008* (Mar. 2008), pp. 181–184. (Citado en la página 181.)

[57] Kitchen, L. y Malin, J. "The effect of spatial discretization on the magnitude and direction response of simple differential edge operators on a step edge". *Computer Vision, Graphics, and Image Processing* 47, 2 (1989), 243–258. (Citado en la página 177.)

[58] Kosaka, A. y Kak, A. C. "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties". *CVGIP: Image Understanding 56*, 3 (Nov. 1992), 271–329. (Citado en las páginas 10 y 15.)

[59] Kramer, J. y Scheutz, M. "Development environments for autonomous mobile robots: A survey". *Autonomous Robots 22* (2007), 101–132. (Citado en la página 147.)

[60] Krotkov, E. "Mobile robot localization using a single image". En: *Proceedings of the 1989 IEEE International Conference on Robotics and Automation* (1989), vol. 2, pp. 978–983. (Citado en la página 14.)

[61] Lee, J.-S. "Digital image smoothing and the sigma filter". *Computer Vision, Graphics, and Image Processing* 24, 2 (1983), 255–269. (Citado en la página 145.)

[62] Leonard, J. J. y Durrant-Whyte, H. F. *Directed Sonar Sensing for Mobile Robot Navigation*, vol. 175 of *The Springer International Series in Engineering and Computer Science*. Springer, 1992. (Citado en la página 124.)

[63] Lienhart, R. y Maydt, J. "An extended set of haar-like features for rapid object detection". En: *Proceedings of the 2002 International Conference on Image Processing* (2002), vol. 1, pp. I–900–I–903. (Citado en la página 106.)

[64] Lindeberg, T. "Edge detection and ridge detection with automatic scale selection". *International Journal of Computer Vision 30* (1998), 117–156. (Citado en la página 175.)

[65] Longley, P. A.; Goodchild, M. F.; Maguire, D. J. y Rhind, D. W. *Geographical Information Systems and Science*. John Wiley & Sons, 2005. (Citado en las páginas 5 y 23.)

[66] MacKenzie, D. C.; Arkin, R. C. y Cameron, J. M. "Multiagent mission specification and execution". *Autonomous Robots 4* (1997), 29–52. (Citado en la página 147.)

[67] Madsen, C. B. y Andersen, C. S. "Optimal landmark selection for triangulation of robot position". *Robotics and Autonomous Systems 23*, 4 (1998), 277–292. (Citado en la página 15.)

[68] Maeda, E. y Minami, Y. "Steps towards ambient intelligence". *NTT Technical Review 4*, 1 (2006), 50–55. (Citado en la página 3.)

[69] Maybeck, P. S. *Stochastic models, estimation, and control*, vol. 141 of *Mathematics in Science and Engineering*. Academic Press, 1979. (Citado en las páginas 4, 37, 38, 41, 43 y 44.)

[70] McGee, L. A. y Schmidt, S. F. *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*. National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California, 1985. (Citado en la página 43.)

[71] McLachlan, G. J. y Krishnan, T. *The EM Algorithm and Extensions*. Wiley series in probability and statistics. John Wiley & Sons, 1997. (Citado en la página 4.)

[72] Meyer, J.-A. y Filliat, D. "Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies". *Cognitive Systems Research*, 4 (2003), 283–317. (Citado en la página 4.)

[73] Mirats Tur, J. M.; Zinggerling, C. y Corominas Murtra, A. "Geographical information systems for map based navigation in urban environments". *Robotics and Autonomous Systems 57* (2009), 922–930. (Citado en la página 12.)

[74] Monasterio, I.; Lazkano, E.; Rañó, I. y Sierra, B. "Learning to traverse doors using visual information". *Mathematics and Computers in Simulation 60*, 3 (2002), 347–356. (Citado en la página 13.)

[75] Montemerlo, M.; Roy, N. y Thrun, S. "Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit". En: *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2003), vol. 3, pp. 2436–2441. (Citado en la página 148.)

[76] MONTEMERLO, M.; ROY, N. Y THRUN, S. "Carmen robot navigation toolkit". http://carmen.sourceforge.net, Julio 2012. (Citado en la página 148.)

[77] MORI, H. Y KOTANI, S. "Robotic travel aid for the blind: Harunobu-6". En: *Proceedings of the Second European Conference on Disability Virtual Reality and Associated Technologies* (1998), pp. 193–202. (Citado en la página 11.)

[78] MUÑOZ, A. J. Y GONZÁLEZ, J. "Two-dimensional landmark-based position estimation from a single image". En: *Proceedings of the 1998 IEEE International Conference on Robotics and Automation* (1998), vol. 4, pp. 3709–3714. (Citado en la página 10.)

[79] MUÑOZ-SALINAS, R.; AGUIRRE, E.; GARCÍA-SILVENTE, M. Y GONZÁLEZ, A. "Door-detection using computer vision and fuzzy logic". *WSEAS Transactions on Systems 10*, 3 (2004), 3047–3052. (Citado en la página 13.)

[80] MURILLO, A. C.; KOŠECKÁ, J.; GUERRERO, J. J. Y SAGÜÉS, C. "Visual door detection integrating appearance and shape cues". *Robotics and Autonomous Systems 56*, 6 (2008). (Citado en la página 13.)

[81] NEIRA, J.; RIBEIRO, M. I. Y TARDÓS, J. D. "Mobile robot localization and map building using monocular vision". En: *Proceedings of the 5th International Symposium on Intelligent Robotic Systems* (Stockholm, Sweden, 1997), pp. 275–284. (Citado en la página 15.)

[82] NEIRA, J. Y TARDÓS, J. D. "Data association in stochastic mapping using the joint compatibility test". *IEEE Transactions on Robotics and Automation 17*, 6 (Dic. 2001), 890–897. (Citado en la página 127.)

[83] NEIRA, J.; TARDÓS, J. D.; HORN, J. Y SCHMIDT, G. "Fusing range and intensity images for mobile robot localization". *IEEE Transactions on Robotics and Automation 15*, 1 (1999), 76–84. (Citado en la página 15.)

[84] NIXON, M. Y AGUADO, A. *Feature Extraction and Image Processing*. Academic Press, 2002. (Citado en la página 177.)

[85] NODA, I. Y ICHI MEGURO, J. "Data representation for information sharing and integration among rescue robot and simulation". En: *Proceedings of SICE-ICASE International Joint Conference* (Oct. 2006), pp. 3449–3454. (Citado en la página 12.)

[86] PAPAGEORGIOU, C. Y POGGIO, T. "A trainable system for object detection". *International Journal of Computer Vision 38* (2000), 15–33. (Citado en las páginas 105 y 106.)

[87] PAPERT, S. "The summer vision project". Informe técnico, Massachusetts Institute of Technology, 1966. (Citado en la página 5.)

[88] PAPOULIS, A. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill Series in Electrical Engineering. McGraw-Hill, 1991. (Citado en la página 39.)

[89] PINTO, N.; COX, D. D. Y DICARLO, J. J. "Why is real-world visual object recognition hard?". *PLoS Computational Biology 4*, 1 (Enero. 2008), 0151–0156. (Citado en la página 5.)

[90] Rackliffe, N.; Yanco, H. A. y Casper, J. "Using Geographic Information Systems (GIS) for UAV landings and UGV navigation". En: *IEEE Conference on Technologies for Practical Robot Applications (TePRA)* (Apr. 2011), pp. 145–150. (Citado en la página 12.)

[91] Rich, S. y Davis, K. H. "Geographic information systems (GIS) for facility management". Informe técnico, IFMA Foundation, 2010. (Citado en la página 5.)

[92] Rusu, R. B.; Meeussen, W.; Chitta, S. y Beetz, M. "Laser-based perception for door and handle identification". En: *Proceedings of the International Conference on Advanced Robotics* (Junio 2009), pp. 1–8. (Citado en la página 14.)

[93] Saffiotti, A. y Broxvall, M. "Peis ecologies: Ambient intelligence meets autonomous robotics". En: *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies* (New York, NY, USA, 2005), sOc-EUSAI '05, ACM, pp. 277–281. (Citado en la página 4.)

[94] Schwan, K.; Eisenhauer, G. y Gu, W. "Cthreads – a parallel programming library". http://www.cc.gatech.edu/systems/projects/Cthreads/, Julio 2012. (Citado en la página 148.)

[95] Shekhar, S. y Chawla, S. *Spatial Databases: A Tour*. Prentice Hall, 2003. (Citado en la página 21.)

[96] Shi, L.; Kodagoda, S. y Dissanayake, G. "Laser range data based semantic labeling of places". En: *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Oct. 2010), pp. 5941–5946. (Citado en la página 12.)

[97] Shi, W. y Samarabandu, J. "Investigating the performance of corridor and door detection algorithms in different environments". En: *International Conference on Information and Automation* (2006), pp. 206–211. (Citado en la página 13.)

[98] Siciliano, B. y Khatib, O., Eds. *Springer Handbook of Robotics*. Springer, 2008. (Citado en la página 37.)

[99] Siegwart, R. y Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004. (Citado en la página 124.)

[100] Simmons, R. G. "Structured control for autonomous robots". *Robotics and Automation, IEEE Transactions on 10*, 1 (Feb. 1994), 34–43. (Citado en la página 148.)

[101] Simmons, R. G. "Inter Process Communication (IPC)". http://www.cs.cmu.edu/afs/cs/project/TCA/www/ipc/index.html, Julio 2012. (Citado en la página 148.)

[102] Sugihara, K. "Some location problems for robot navigation using a single camera". *Computer Vision, Graphics, and Image Processing 42*, 1 (1988), 112–129. (Citado en la página 14.)

[103] Tardós, J. D. "Representing partial and uncertain sensorial information using the theory of symmetries". En: *Proceedings of the 1992 IEEE International Conference on Robotics and Automation* (Mayo 1992), vol. 2, pp. 1799–1804. (Citado en la página 15.)

[104] Thrun, S. "Robotic mapping: A survey". En *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002, pp. 1–36. (Citado en las páginas 4 y 16.)

[105] Thrun, S.; Beetz, M.; Bennewitz, M.; Burgard, W.; Creemers, A.; Dellaert, F.; Fox, D.; Hahnel, D.; Rosenberg, C.; Roy, N.; Schulte, J. y Schulz, D. "Probabilistic algorithms and the interactive museum tour-guide robot minerva". *International Journal of Robotics Research 19*, 11 (Nov. 2000), 972–999. (Citado en la página 4.)

[106] Thrun, S.; Burgard, W. y Fox, D. *Probabilistic Robotics*. MIT Press, Cambridge, Ma, 2005. (Citado en las páginas 4, 37, 43 y 44.)

[107] Tsai, R. Y. "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses". *IEEE Journal of Robotics and Automation 3*, 4 (Ago. 1987), 323–344. (Citado en la página 76.)

[108] Vasudevan, S.; Gächter, S.; Nguyen, V. y Siegwart, R. "Cognitive maps for mobile robots – an object based approach". *Robotics and Autonomous Systems 55*, 5 (Mayo 2007), 359–371. (Citado en las páginas 4 y 92.)

[109] Viola, P. y Jones, M. J. "Robust real-time face detection". *International Journal of Computer Vision 57*, 2 (2004), 137–154. (Citado en las páginas 106, 181 y 182.)

[110] Vosniakos, G.-C. y Mamalis, A. "Automated guided vehicle system design for fms applications". *International Journal of Machine Tools and Manufacture 30*, 1 (1990), 85–97. (Citado en la página 9.)

[111] Weiser, M. "The computer for the 21st century". *Scientific American 265*, 3 (1991), 66–75. (Citado en la página 3.)

[112] Weiss, G.; Wetzler, C. y von Putkammer, E. "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans". En: *Proceedings of the International Conference on Intelligent Robots and Systems* (1994). (Citado en las páginas 96 y 99.)

[113] Yang, X. y Tian, Y. "Robust door detection in unfamiliar environments by combining edge and corner features". En: *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (Junio 2010), pp. 57–64. (Citado en la página 13.)

[114] Yeung, A. K. y Hall, G. B. *Spatial Database Systems*, vol. 87 of *GeoJournal Library*. Springer, 2007. (Citado en la página 29.)

[115] Yoshida, T.; Nagatani, K.; Koyanagi, E.; Hada, Y.; Ohno, K.; Maeyama, S.; Akiyama, H.; Yoshida, K. y Tadokoro, S. "Field experiment on multiple mobile robots conducted in an underground mall". En *Field and Service Robotics*, A. Howard, K. Iagnemma, and A. Kelly, Eds., vol. 62 of *Springer Tracts in Advanced Robotics*. Springer Berlin / Heidelberg, 2010, pp. 365–375. (Citado en la página 12.)

[116] ZARCHAN, P. Y MUSOFF, H. *Fundamentals of Kalman Filtering: A Practical Approach*, vol. 208 of *Progress in Astronautics and Aeronautics*. American Institute of Aeronautics and Astronautics, 2005. (Citado en la página 43.)

VNiVERSiDAD
Đ SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

COLOFÓN

Esta tesis ha sido compuesta con LaTeX 2ε utilizando la fuente *Bera Mono*, desarrollada originalmente por Bitstream, Inc. como "Bitstream Vera". (Las fuentes PostScript Tipo 1 han sido proporcionadas por Malte Rosenau y Ulrich Dirr.)

El estilo tipográfico se ha inspirado en el brillante trabajo de Bringhurst presentado en *The Elements of Typographic Style* [19]. Se puede descargar gratuitamente para LaTeX via CTAN como "`classicthesis`".