UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería de Sonido e Imagen en Telecomunicación

Trabajo de Fin de Grado

Descripción Multi-Capa del entorno mediante información de curvatura para navegación robótica

---

Autor:
Javier Almeida Gutiérrez

Supervisor:
Pedro Núñez Trujillo

Noviembre, 2014

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Grado en Ingeniería de Sonido e Imagen en Telecomunicación

Trabajo de Fin de Grado

Descripción Multi-Capa del entorno mediante información de curvatura para navegación robótica

---

Autor:
Javier Almeida Gutiérrez
Fdo:

Supervisor:
Pedro Núñez Trujillo
Fdo:

Tribunal Calificador:

Presidente:

.....

Fdo:

Vocal:

.....

Fdo:

Secretario:

.....

Fdo:

CALIFICACIÓN:

Noviembre, 2014

Agradecimientos

En primer lugar, empezar por agradecer a todas aquellas personas que de mayor o menor manera han colaborado en que este proyecto sea posible.

Comenzando por el equipo de RoboLab, que han contribuido en varias facetas complementarias al trabajo principal y que me han recibido de buen agrado. Concretamente dar las gracias al tutor de éste proyecto Pedro Núñez y a Luis Manso, por permitirme continuar con parte de su trabajo, por la dedicación y el interés prestado.

Para finalizar, todas aquellas personas que siempre están y que me aportan la fuerza y el apoyo. A toda mi familia y especialmente a mi hermana Marta.

A los que siempre están...

Contenido

Acrónimos

CUBA	CUrvature Based Algortihm
SLAM	Simulation Localization And Mapping
UEX	Universidad de Extremadura
EPCC	Escuela Politécnica de Cáceres
TFG	Trabajo de Fin de Grado
RCIS	RoboComp InnerModel Simulator
WAF	Workshop of Physical Agents
COP	Components Oriented Programming
HAL	Hadware Abstraction Layer

Índice de figuras

Índice de cuadros

Resumen

Nos encontramos en un capítulo de la historia donde los avances tecnológicos y la investigación son, o deberían ser, una de las principales prioridades del ser humano. La robótica autónoma es objeto de investigación desde hace algunas décadas y siguen apareciendo avances continuamente, y mucho queda aún por investigar. Concretamente el ámbito de la navegación autónoma es un campo en el que mucho se ha investigado y se seguirá perfeccionando. ¿Cuál será la manera más eficiente para que un robot sea independiente para desplazarse? Esa es la pregunta que se intenta responder.

Este trabajo presenta novedoso método para las descripción de entornos multi-capa basado en la información de curvatura de los alrededores del robot. Se parte de un método desarrollado anterior llamado *CUrvature Based Algortihm (CUBA)*, el cuál consiste a grandes rasgos, en una descripción del entorno mono-capa proyectando por tanto, las diferentes características de los alrededores del robot en un plano 2D a una altura determinada.

El método consiste en 3 fases consecutivas: Adquisición de datos y pre-procesado, segmentación y extracción de puntos de referencia, a las que hay que añadirles la novedad principal, que es el uso de un sensor RGBD (*Kinect*) el cual permite obtener las características del entorno en 3D, considerando un conjunto de planos situados a diferentes alturas.

Cada uno de estos planos corresponde a diferentes *scans* lasers virtuales, que son obtenidos a partir del sensor RGBD. A continuación dichos *scans* son segmentados utilizando una estimación adaptativa de curvatura o **función de curvatura**. Esta segmentación adaptativa se utiliza directamente para extraer las características del entorno que se diferenciaría entre *puntos de ruptura, segmentos de linea*

y esquinas reales y virtuales de los diferentes planos. Por último toda esta información se sincroniza en tiempo real agrupándolas en el entorno 3D.

Este proyecto será respaldado con resultados experimentales realizados tanto en entornos reales como en entornos simulados, a partir de los cuales se comprueba que el método expuesto es eficaz para detectar puntos de referencia en estructuras y semi-estructuras, entendiendo como tales, las paredes que limitan un recinto y los objetos que puedan presentarse en el entorno respectivamente. Por tanto estas experiencias compondrán un estudio comparativo en términos de robustez y eficiencia que expone las mejoras del sistema de descripción de entorno que se presenta.

Palabras Claves

conjunto láser virtual, función de curvatura adaptativa, descripción de entorno 3D, marcas naturales

Capítulo 1

Introducción

El primer paso en el desarrollo de un proyecto, sería el planteamiento de un problema. Este Trabajo de Fin de Grado (TFG) se centra en la robótica autónoma, la cual se define como: "*el área de la robótica que desarrolla robots capaces de desplazarse y actuar sin intervención humana*". A lo largo de la literatura se han desarrollado muchos métodos para la navegación autónoma, y todos parten del mismo punto, ya que un robot autónomo trabajando en escenarios reales debe tener la capacidad de "navegar" en este entorno y por tanto uno de los pre-requisitos necesarios será que el robot pueda construir un mapa o representación del área de trabajo (*mapping*).

1.1. Motivación de este TFG

Como hemos dicho, todo proyecto de investigación tiene como objetivo y motivación para el investigador, resolver un problema. La finalidad principal de este proyecto consiste en extraer la información espacial para describir las características del entorno, lo cual ofrece un importante efecto positivo sobre muchos problemas presentes en la robótica:

- Posicionamiento y localización del robot en cualquier entorno de trabajo. Se puede definir como el problema básico en el ámbito de la navegación autónoma, y engloba 3 objetivos fundamentales: El **primero** consiste en la localización concreta del robot en el entorno en el que se mueve, punto de partida para cualquier operación que requiera el movimiento del robot.

Esto nos lleva al **segundo** objetivo, que consiste en el destino que tomará el robot al emprender una acción (aún no se ha investigado mucho en este aspecto ya que suele venir dado por el usuario del robot). Por último el **tercer** y último objetivo consistiría en el proceso que debe llevar a cabo el robot para planificar la ruta que tomará el robot para ir desde un punto inicial a un punto destino. Dicha ruta dependerá por tanto del destino que se haya definido y esta técnica se puede enfocar de muchas maneras distintas (Rutas locales, generales, etc). En el mundo de la robótica a este proceso se denomina *Path planing* o *Motion planing*.

- Evitación de obstáculos, ya que cualquier robot en cualquier entorno de trabajo se encontrará obstáculos de todos los tamaños y formas. Dicho robot deberá tener la capacidad de reconocerlos y evaluarlos según convenga. Este problema también esta englobado en *Motion Planing* puesto que a la hora de trazar cualquier ruta se presentarán obstáculos. La planificación general evitaría principalmente los obstáculos de dimensiones mayores y en las planificaciones locales se evitarían los obstáculos de menor envergadura, una planificación más "fina" y detallada.

- SLAM o *Simulation Localization And Mapping*. Esta técnica es muy importante dentro del mundo de la robótica pues emplea a la vez la construcción de un mapa del entorno no conocido y todos los obstáculos que podría contener, y la planificación de la ruta por donde se desplazará dicho robot. Esta técnica une por tanto, tanto *Motion Planning* como la creación de dicho mapa (Un mapa 3D en este caso), e investiga los problemas que surgen a la hora de crear un modelo matemático, geométrico o lógico del entorno físicos empleando un robot autónomo. Concretamente el SLAM tiene como objetivo solucionar los problemas a la hora de colocar un robot autónomo en un entorno desconocido, situándose y trazando rutas a partir del mapa físico que el mismo robot crea mediante sensores.

El método presentado esta ligado y surge como solución directa o indirecta para estos problemas. Hablamos de solución directa e indirecta ya que este método se ocupa de la parte principal del problema que consiste en la descripción del

entorno. Para esta descripción un robot debe ser equipado con herramientas que le permitan percibir la información del entorno al igual que los humanos están equipados de sus sentidos. Los "sentidos" de un robot serian los sensores con los que se equipa, en nuestro caso el sensor utilizado va ser el RGBD, que es una de las principales novedades de este nuevo método.

1.2. Tema y marco de trabajo de este TFG

En teoría, si un robot es equipado con sensores puede adquirir una gran cantidad de información sobre la estructura espacial de un entorno desconocido para el robot. Existen cantidad de sensores que se incorporan a los robots y que *perciben la información de las magnitudes físicas del entorno transformándolas después en señales eléctricas que sistema de control pueda procesar*. Para navegación la magnitud física que requerimos evidentemente es la distancia a las estructuras o semi-estructuras que componen el entorno.

Tradicionalmente se han utilizado sensores de visión o rango por la mayoría de los investigadores en las últimas décadas, nos referimos a dos tipos principalmente entre otros, los **sonar** y los **lasers**. Los primeros nos permitían conocer la distancia a los objetos del entorno a partir de los ultra-sonidos que se reflejaban en estos, mientras que los lasers utilizaban rayos infrarrojos con la misma función de reflexión.

Recientemente los sensores **RGBD de bajo coste** han tenido gran repercusión en la comunidad robótica, dado que estos sensores RGBD nos ofrecen a la vez información de imagen RGB e información de distancia mediante su sensor de profundidad. Esta es una de las principales contribuciones de este proyecto, percibir la distancia a las estructuras del entorno a partir de dicho sensor lo que permitirá además muchas más funcionalidad del robot al tener cámara y sensor de profundidad en el mismo *hardware*. Independientemente del tipo de sensor, el robot autónomo debe elegir la representación espacial y ser capaz de procesar la lectura en tiempo real.

Este proyecto toma un enfoque basado en marcas naturales del entorno y sus

estructuras para la representación espacial utilizando un sensor RGBD, donde podemos definir **marcas** como: *Características del entorno que un robot puede distinguir fiablemente a partir de la observación sensorial.* Como regla general, que un sistema de descripción de entorno para navegación autónoma tenga éxito depende de dos puntos esenciales:

- El tipo elegido de características, y la existencia de sensores precisos capaces de adquirir suficientes características para una descripción completa del entorno.
- La disponibilidad de algoritmos rápidos y seguros capaces de extraer y caracterizar marcas dentro un gran conjunto de ruido y de datos inciertos.

1.3. Principal contribución de este TFG

La **principal contribución** de este proyecto sería proveer al robot con la capacidad para extraer varios tipos de marcas, que serán presentados en estructuras o semi-estructuras 3D en entornos de interior utilizando sensores RGBD en tiempo real.

Toda esta información adquirida por el sensor RGBD se agrupa en un conjunto de planos situados a distintas alturas según la configuración y a continuación las información de cada plano es segmentada utilizando la estimación adaptativa de curvatura. Por último se extraen y se clasifican las marcas de cada plano (Figura 1.1).
Este proyecto extiende el trabajo presentado por los autores en [1], mejorando el mapa generado por el robot e incluyendo más características en el entorno 3D del robot para ser utilizadas en futuros trabajos como SLAM o métodos de exploración.

1.4. Organización de este TFG

Esté TFG es el resultado del esfuerzo de muchas personas y no solo del autor principal, comenzará por tanto con una sección de agradecimientos y una dedi-

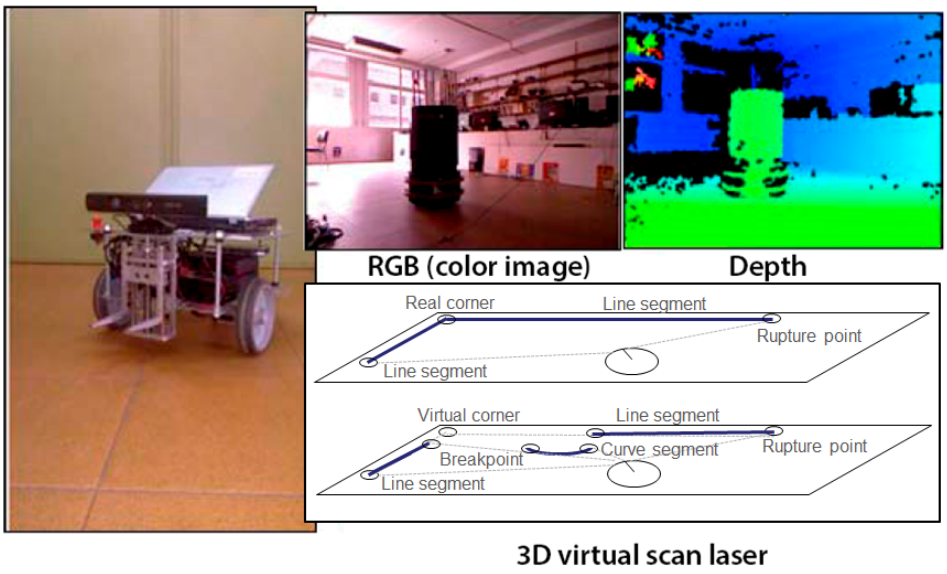

Figura 1.1: El nuevo enfoque utiliza el sensor RGBD montado en el robot RobEx para extraer y caracterizar un conjunto de marcas naturales: Segmentos rectos y curvos, y esquinas reales y virtuales. La información RGBD es proyectada en diferentes planos y cada uno es utilizado como un *scan* virtual láser

catoria. EL siguiente conjunto de secciones formarán el preámbulo de este TFG, el cual está iniciado por un índice de contenidos bastante completo, donde se indican todos los capítulos de este proyecto con la respectiva página en la que comienzan, y que además permite navegar por el contenido en la versión *pdf*. A continuación se encuentran una serie de listas donde se enumeran todas las figuras, acrónimos, tablas y ecuaciones que aparecen en la documentación de este TFG y para finalizar el preámbulo un pequeño resumen en lengua castellana y un resumen más extenso en lengua inglesa, donde se explican las ideas principales y el desarrollo en general de este método. El cuerpo del documento están formado por 7 capítulos y 3 apéndices.

Capítulo 2

Literatura relativa

2.1. Introducción

Para poder desarrollar una proyecto de investigación y poder realizar un estudio comparativo respecto a otros proyectos hay que tener en cuenta el trabajo relativo en el ámbito en el que se quiere investigar. Como hemos ya introducido, éste proyecto está centrado en el ámbito de la robótica autónoma y en particular en la navegación del robot. Este capítulo presenta el actual **estado del arte** para el problema de la representación de entorno para robots autónomos y la intervención de los sensores RGBD en este campo. La primera sección comentará los algoritmos y enfoques más determinantes en la literatura, explicando el enfoque basado en *marcas naturales*, que es el enfoque en el que se basa el algoritmo CUBA. Por último se analizarán los diferentes sensores utilizados para representar el mapa y las ventajas de utilizar un sensor RGBD.

Extraer la información útil a partir del entorno del robot tiene un efecto muy importante en el proceso de navegación del robot puesto que muchas aplicaciones requerirán información detallada del mismo (por ejemplo, SLAM). Pero esta tarea requiere de un proceso complicado en el cual intervienen muchos condicionantes físicos que influirán a la hora de elegir un modelo de representación:

1. El primer condicionante será el **tipo de entorno** a considerar, el cual ha sido tratado en la literatura desde varias situaciones. Un robot autónomo deberá desarrollar acciones en entornos muy variados, estos podrían ser *es-*

táticos lo cual es una valoración muy ideal, o *dinámicos* donde el entorno es cambiante y se acerca más a la realidad. Independientemente de ser un entorno estático o dinámico habrá que valorar si es un entorno *estructurado* o *sin estructurar*, ya que ello influirá drásticamente en la movilidad del robot. También se diferenciará entre entornos de *interior* o de *exterior* y como afecta ello al robot. Por último tener en cuenta el tamaño del entorno, diferenciando entre *pequeña escala* y *gran escala*, lo cual es una de las mayores limitaciones a la hora de desarrollar un algoritmo de navegación.

2. El segundo consiste en el **tipo de sensor** que se utilizará, lo cual es una de los principales condicionantes a la hora de elegir el modelo y también de los más importantes, puesto que determinará la calidad de la representación. El sensor más apropiado dependerá de 3 factores principalmente: El tamaño del área de operación, las condiciones del entorno y el nivel de representación requerido. En la figura 2.1 podemos ver 3 ejemplos de los sensores más utilizados.

3. El tercer y último condicionante reside en el **tipo de marcas**. El concepto de *marca* es el elemento utilizado para describir el entorno y dependerá del enfoque que utilicemos para ello. A lo largo de la literatura se han clasificado los principales enfoques de la siguiente manera:

   - Enfoques **Cuantitativos o métricos**: Donde la representación viene dada por información métrica adquirida por los sensores. En esta clasificación entraría el método en el que se basa el algoritmo CUBA, que es un enfoque tipo *features-based* [1]. También estaría el método basado en rejilla, *grid-based* [3].
   - Enfoques **Cualitativos**: Se basan en dividir los espacios dentro de los límites de las estructuras del entorno intentando imitar los procesos cognitivos del hombre. En está clasificación los enfoques típicos en la literatura serían el modelo topológico, *topologic-based* [2], y el modelo basado en semántica, *semantic-based* [4].
   - Enfoques **Híbridos**: Combinan los mejores enfoques cualitativos y cuantitativos.

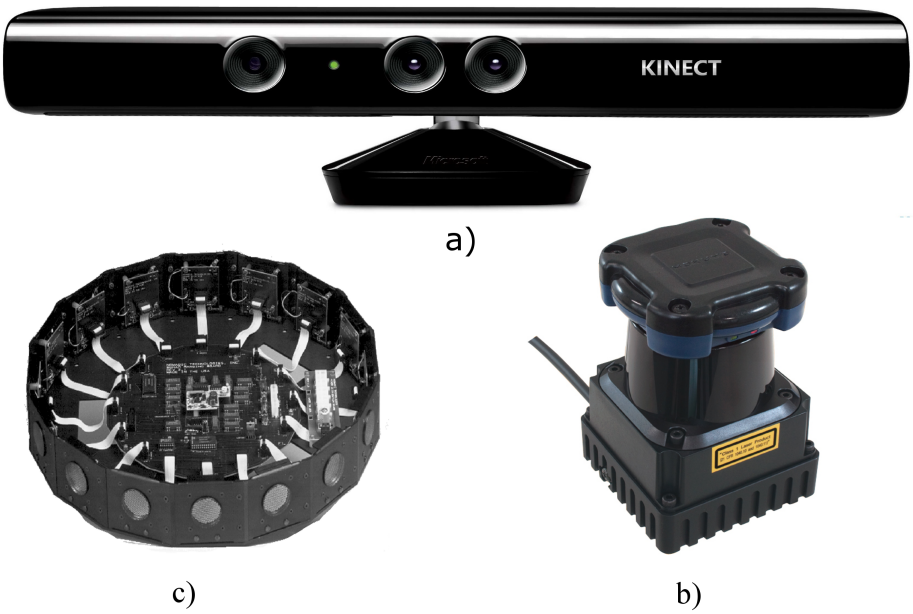

Figura 2.1: En el apartado a) se puede ver un sensor tipo RGBD como el utilizado en el método propuesto y en la figura b) un típico telémetro de rango laser, sensor utilizado en el anterior trabajo. En la figura c) se muestra un anillo sonar que consta de 16 sensores sonar.

En este proyecto como ya hemos comentado, se utiliza un enfoque basado en marcas del entorno, concretamente en la estimación adaptativa de curvatura, aunque en este capítulo analizaremos también otro modelos. Además de esta clasificación principal también se han desarrollado enfoques menos importantes, como vemos en [5]. En resumen, en esté capitulo se analizará la elección de un modelo representativo (teniendo en cuenta los condicionantes) y las principales contribuciones al problema de descripción de entorno.

2.2. Representación del robot móvil en su entorno de trabajo

Como se ha mencionado en la introducción, el problema de la representación del robot móvil en su entorno ha sido afrontado mediante varios tipos de enfoque y se han clasificado según el tipo solución que nos ofrecen. En esta sección se van a comentar dichos enfoques según esta clasificación, comenzando por los **cuanti-**

tativos, los cuales generan un mapa local del entorno a partir de la información métrica obtenida por el sensor o sensores con el que el robot es equipado.

2.2.1. Grid-Based

Son los principales modelos cuantitativos, **los mapas métricos** considerados son bi-dimensionales y consisten en dividir el entorno en rejillas de celdas, las cuales tienen asignado un valor (En la Figura 2.2 se puede apreciar un ejemplo). Este enfoque ha sido utilizado en varios sistemas desde que fue propuesto por primera vez con resultados satisfactorios y explicado brevemente funciona de la siguiente manera: Cada celda de la rejilla (x, y) en el mapa tiene un valor de ocupación asignado el cuál indica la probabilidad de estar ocupada por algún obstáculo. Si la celda correspondiente estuviera ocupada se incrementaría su valor. Para construir un mapa basado en rejilla se siguen por lo general 4 fases principales:

1. *Interpretación del sensor.* Se asigna el valor de ocupación a las celdas.
2. *Integración en tiempo.* Se integra la información de varios sensores para conseguir una única estimación de ocupación.
3. *Estimación de ocupación.* Se corrigen continuamente errores de odometría.
4. *Exploración.* El robot se mueve a través de los espacios libres hacia terreno inexplorado.

Realmente solo las dos primeras fases pertenecen al ámbito de la representación del entorno puesto que la tercera y la cuarta se centran en las aplicaciones de *mapping* y por tanto no las analizaremos. La etapa de interpretación se relaciona con el mapa local obtenido a partir del análisis del entorno, y se desarrolla con diferentes técnicas, siendo las más populares *Las reglas de Bayes*. Respecto a la fase de integración, en la literatura, se ha comprobado que una lectura de la gama completa solo ofrece una pequeña cantidad de información y por tanto, para obtener un mapa del entorno preciso es necesario combinar la información de muchas lecturas.

Los métodos basados en rejillas de celdas de ocupación utilizando información

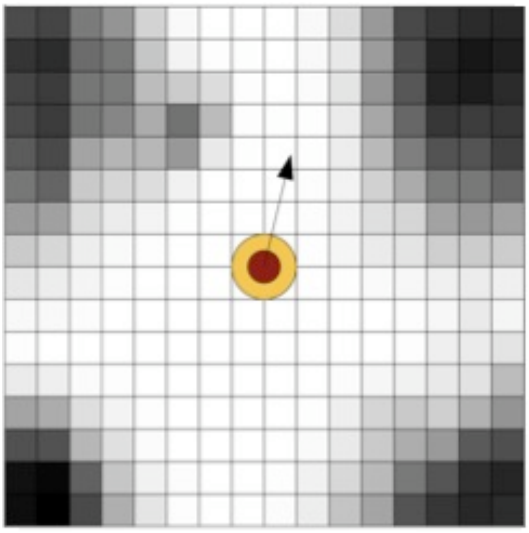

Figura 2.2: En la figura se puede apreciar un esquema de un robot movil desarrollando un mapa del entorno mediante un descriptor basado en el método de rejilla. El tono más o menos oscuro de las celdas representa el valor de probablildad de estar ocupadas

obtenida mediante sonar, telémetros láser y sensores de visión puede proveer un mapa local del entorno bastante robusto, aunque en este proyecto se utilizará el enfoque basado en marcas como veremos a continuación, ya que el basado en rejilla tiene ciertas ventajas y desventajas. Una de sus **ventajas** es su facilidad para computar rutas cortas y reconocimiento de lugares, aunque el precio de estas capacidades se paga ya que es un enfoque computacionalmente caro, que requiere una enorme cantidad de memoria y donde precisión del mapa generado depende del tamaño de la celda que consideremos en la fase de diseño.

2.2.2. Features-Based

Otro enfoque de carácter cuantitativo serían los modelos de representación del entorno basados en marcas del entorno, considerando éstas como un conjunto de características que encontramos comúnmente en el entorno, tales como, lineas, círculos, etc. (por ejemplo, en primeras investigaciones sobre este enfoque se utilizaron poliedros). Las marcas y el método para su caracterización que utilicemos para delimitar el entorno de alrededor del robot dependerá de muchos factores, como el entorno (interior o exterior) y el tipo de sensor, por eso en esta sección nos centraremos en algoritmos basados en sensores láser, ya que aunque nosotros utilizaremos un sensor RGBD como ya hemos introducido, a partir de está in-

formación se generarán un conjunto de lásers virtuales, y son los más utilizados para aplicaciones de navegación.

Estos sensores, tanto los telémetros láser como los sonar, obtienen una representación en 2D (plano x-y) del entorno basado en la distancia a los objetos que rodean al robot. Esta representación se basa en utilizar un grupo de marcas o características, de manera que las estructuras o semi-estructuras de los entornos de interior pueden ser modeladas utilizando modelos geométricos como son *segmentos rectos*, *segmentos curvos*, *esquinas* o todos a la vez (En la Figura 2.3 se muestra una porción de un entorno de interior descrita mediante el algortimo CUBA). Estas marcas se tratarían siguiendo modelos matemáticos, de manera que para los segmentos rectos se definirían su longitud, posición y orientación, para las esquinas su posición y orientación y para los círculos su posición y su radio. Es por esto, que para el desarrollo de un algoritmo eficiente es necesario el conocimiento de estos modelos matemáticos, siendo utilizadas dos técnicas en pruebas iniciales en la literatura:

- *Clustering* es decir, **la transformada de Hough**: Es una técnica robusta para encontrar lineas en un conjunto de puntos en 2D. Esta basado en una transformación del plano Cartesiano (x, y), al dominio de Hough (ρ, θ).
- **Ajuste por mínimos cuadrados**: Al adquirir la información de los datos en bruto suelen venir acompañados de ruido, es por esto que una estimación de la tendencia de los resultados se hace necesaria (también llamado ajuste de la curva).

Dado un rango de lectura, muchos algoritmos han sido propuestos para el mapeo en la robótica móvil utilizando segmentos rectos extraídos de las diferentes lecturas láser en 2D. El trabajo presentado en [8] propone un interesante estudio y comparación sobre los algoritmos de extracción de lineas para entornos de interior, incluyendo:***Split-and-Merge*, *Incremental*, *Hough-Transform*, *Line regresion*, *Random Sample Consensus (RANSAC)*, *ExpectationMaximization (EM)***

Recientemente, en [9], *Distance-Based Convolution Clustering (DCC)* fue introducida para agrupar los puntos escaneados por los robots en algunos *clusters*

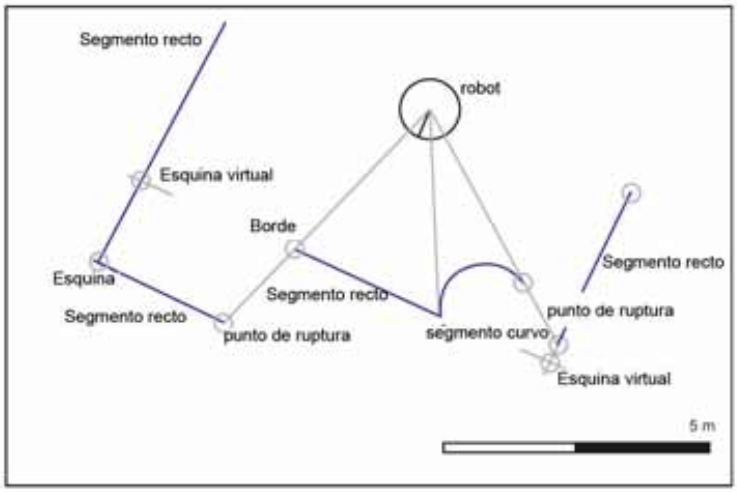

Figura 2.3: En la figura se aprecia un entorno de interior descrito a partir de las diferentes marcas que utiliza el algortimo CUBA para caracterizarlo. Se verá de manera extensa en el Capítulo 5 de este TFG

utilizando la operación de convolución. Estos *clusters* fueron utilizados para detectar lineas con una combinación de la Transformada de Hough y el algoritmo de seguimiento de linea.

En *Mohamed et al.'s work* [**?**], los autores definen *Consecutive Clustering Algorithm (CCA)* que añade nuevas lineas incrementalmente a los mapas de lineas calculados previamente y los fusiona con el mapa general de acuerdo a un análisis basado en estadística. Nuevos enfoques como el presentado en [11], adopta una nueva metodología para detectar características lineales y circulares que no dependen del conocimiento previo del entorno.

Todos estos enfoques previos procesan las lecturas del láser en 2D para encontrar un conjunto de características geométricas, pero hay otros enfoques tales como el presentado en [12], o el más reciente en [1], los cuales analizan la lectura láser como un descriptor local el cual puede ser procesado para extraer un conjunto dominante de puntos cuyas lecturas correctamente segmentadas dividen el entorno en segmentos rectos y segmentos curvos. En estos enfoques, en lugar de utilizar una solución lenta e iterativa, los puntos dominantes son robustamente detectados mediante la adaptación de la escala para los entornos locales de cada lectura de la distancia.

El trabajo presentado en este proyecto utiliza el enfoque de *adaptative curvature* descrito en [1], que permite segmentar robustamente las lecturas de los lásers virtuales (es decir, proyecciones de la información de profundidad en los distintos planos en 2D) en segmentos curvos, segmentos rectos y esquinas (reales y virtuales) de una forma rápida y que veremos como se aplica para este proyecto en el Capítulo 5.

Este enfoque tiene como ventaja que permite describir las diferentes partes del entorno del robot mediante un gran conjunto de elementos geométricos. En general, estos métodos reducen drásticamente la carga computacional y las restricciones de memoria, y los hace atractivo gracias a su versatilidad y compacidad.

2.2.3. Topological-Based and Semantic-Based

Estos enfoques son clasificados como **cualitativos** como se mencionó en la introducción. Emplean **los mapas topológicos** los cuales son construidos superponiéndose a los mapas métricos de rejilla. La idea es sencilla y efectiva: El espacio libre del mapa de rejilla se divide en un pequeño número de regiones, separadas por *Critical Lines*. Estas lineas críticas corresponden a pasos pequeños del entorno tales como puertas. Por último a partir del mapa particionado se crea un grafo en donde cada nodo es una región correspondiente al mapa topológico (En la Figura 2.4 se puede apreciar un ejemplo). **El mapa semántico** sería el siguiente paso, el cual consiste en asignar una semántica a cada región del mapa topológico (por ejemplo, salón, cocina, aseo, etc).

Estos mapas permiten realizar rutas bastante eficientes pero su **desventaja** es que se basan en escenarios predefinidos donde se conocen de antemano alguna de las estructuras o semi-estructuras del entorno (por ejemplo, pasillos o habitaciones).

2.3. Estado actual del arte del sensor RGBD

Como ya hemos explicado en las secciones anteriores, este proyecto utiliza un enfoque basado en marcas y por tanto se analizarán los sensores utilizados en la

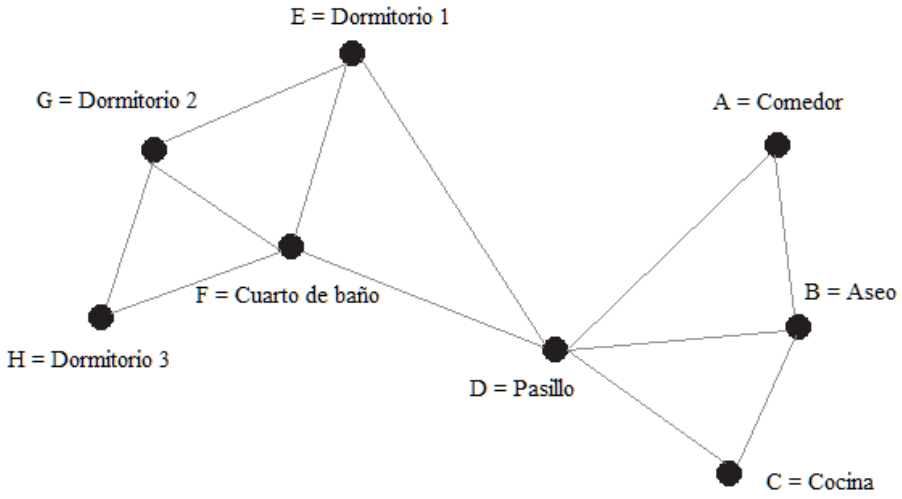

Figura 2.4: En esta figura se puede apreciar un grafo creado a partir del método topológico, y añadiendo a cada nodo diferente una semántica. El entorno es descrito como se aprecia entre nodos (lugares) y relaciones (rutas). Se suele utilizar para trazar rutas cortas entre nodos.

literatura mediante este enfoque. La mayoría de los enfoques basados en marcas se basan en estructuras cuya naturaleza difiere de acuerdo con el medio ambiente (por ejemplo, en interiores o al aire libre), y el tipo de sensor con el que el robot está equipado. Sensores basados en visión se han utilizado para extraer las diferentes características del entorno del robot (Se hace un buen enfoque en [**?**]), sin embargo el incluir la visión para resolver el problema del mapeo agrava el uso de CPU debido a la complejidad de sus algoritmos. Sensores de rango tales como el sonar o el telémetro láser, han sido también utilizados para extraer características del entorno ([1], [5]). Contrario a los sistemas de visión, los algoritmos implementados para trabajar con láser y sonar son normalmente menos complejos.

Finalmente, los sensores RGBD permiten al robot describir el entorno utilizando al mismo tiempo información de imagen y de profundidad, procesando nubes de color de puntos en 3D los cuales tienen una carga computacional muy grande [6]. El enfoque descrito en este proyecto trabaja con información de profundidad unicamente, pero a diferencia de otros métodos, el robot utiliza la información

de profundidad obtenida a partir del sensor RGBD para generar un conjunto de sensores de telemetría láser cuya lectura en 2D es procesada independientemente y de manera paralela.

Capítulo 3

Hipótesis y objetivos

En este capitulo se explicarán los principales objetivos que se han propuesto para este proyecto y las hipótesis planteadas en el comienzo del mismo.

El propósito principal de este proyecto era tratar de contribuir a la navegación autónoma, que dentro de la robótica, es un tema que sigue en auge y más en concreto a los métodos basados en características. Fijamos entonces el punto de partida en otro proyecto anterior, el algoritmo CUBA como ya hemos mencionado en anteriores apartados. La **hipótesis** que da pie a este trabajo fue exportar la descripción del entorno que ofrece el algoritmo CUBA a las 3 dimensiones, lo que supondría un gran avance abriendo un abanico de nuevas posibilidades de aplicación y de continuación de este proyecto en trabajos futuros, además de una descripción de entorno más robusta y eficiente.

La solución para conseguir la descripción del entorno en 3D surgió con la aparición de los nuevos sensores RGBD y su coste abaratado, de manera que no solo obtendríamos una percepción del entorno más global al permitir analizarlo a cualquier altura deseada dentro del campo de visión del sensor, sino que también se pueden añadir características de color de las estructuras que componen el entorno. Con está solución al problema surgen entonces los **objetivos** directos e indirectos de este proyecto que son varios y vamos a explicar:

- El **principal objetivo** consistiría en conseguir procesar la información en bruto obtenida por el RGBD, dividiéndola en un conjunto de lecturas láser virtuales para poder caracterizarlas a través del algoritmo CUBA.

- El segundo objetivo principal sería conseguir agrupar las marcas naturales de cada altura del entorno ya caracterizadas a partir del algoritmo CUBA, para componer el mapa de representación 3D del alrededor del robot.

- Todo este proyecto está englobado en **RoboComp**, el *framework* desarrollado por **RoboLab** y el cual ampliaremos en el Anexo A. A consecuencia de ello surgen objetivos adicionales a los principales, empezando por familiarizarse con dicho marco de trabajo, aprender su funcionamiento y como se compone, y adaptar el algoritmo CUBA a RoboComp.

- El compendio de algoritmos utilizados en este proyecto se han desarrollado en el lenguaje de programación *C++*, lo que provoca de nuevo otro objetivo transversal. Ampliar el conocimiento sobre esté lenguaje y la habilidad de programar en el mismo para poder implementar los algoritmos propuestos, se convierte en competencia básica, para desarrollar este proyecto.

- Un objetivo adicional surge al plantear las experiencias y pruebas que se deben realizar para verificar el correcto funcionamiento y la eficiencia del método propuesto. Dichas experiencias no solo se realizan en entornos del robot reales sino que también se trabaja con entornos simulados, se requiere como competencia transversal comprender la utilización de uno de los componentes de RoboComp denominado **RCSimulator**.

- Como último objetivo adicional. la herramienta comentada en el punto anterior nos permite caracterizar a partir del método propuesto, entornos que serán desarrollados en el lenguaje *XML*, y por tanto ampliar el conocimiento sobre este lenguaje es importante para el desarrollo de los entornos de simulación.

Objetivos principales y objetivos adquiridos al comienzo del desarrollo de este proyecto forman un compendio jerarquizado de propósitos conseguidos, siendo cada uno de ellos competencias necesarias para poder desarrollar un método que contribuya a la navegación autónoma de manera eficaz.

Capítulo 4

Láser virtual en 3D a partir de información RGBD

4.1. Introducción

La primera fase en el proceso de descripción del entorno del robot, sería la captación de la información de profundidad a partir del sensor RGBD como ya hemos introducido. El sensor RGBD transmitirá esta información de profundidad 3D al robot el cual la procesará para crear el conjunto de lásers virtuales. Este conjunto de lásers virtuales nos ofrecerán las lecturas en 2D a las diferentes alturas configuradas y a continuación se procesará esta información de profundidad en 2D mediante la estimación adaptativa por curvatura. La principal ventaja de utilizar un conjunto de lecturas láser virtual reside en la reducción de carga computacional que ofrece al no requerir todos los puntos de profundidad que se obtienen mediante el sensor RGBD. Además se consigue una caracterización del entorno bastante eficiente como se demuestra en el Capítulo 6 "Resultados Experimentales", que permite un gran abanico de aplicaciones futuras.

4.2. Sensor RGBD

¿Que tipo de información tratamos? Como se ha introducido, la primera fase del proceso es la captación de esta información de profundidad e imagen. Dicha información dependerá de las especificaciones del sensor utilizado. En este proyecto

solo se utiliza la información de profundidad aunque la información de imagen podría ser utilizada en futuros trabajos. El sensor transmite por tanto la información del entorno del robot, dentro del campo de visión del sensor, en forma de matriz de puntos discreta, que dependerá de la resolución de dicho sensor. La resolución de imagen RGB y la resolución de distancia puede ser distinta y por tanto el sensor nos proporcionará mucha más información de color que de profundidad.

Por tanto la *point cloud* o nube de puntos de profundidad captados por el sensor RGBD, con la que se tratará en el siguiente apartado, dependerá de 3 especificaciones principalmente:

- **Resolución de matriz de profundidad:** Dicha resolución se define como el número de puntos de profundidad de ($altura \times anchura$). Dividiendo el entorno visto por el sensor en una cantidad discreta de puntos.
- **Campo de visión del sensor:** Este campo visión se define como el factor que determina que parte de la escena es captada. El ángulo de visión depende de la distancia focal, la cual viene directamente dada por el objetivo del sensor.
- **Rango de profundidad:** Se define como los valores de profundidad mínima y máxima que puede detectar el sensor $[D_{min}, D_{max}]$.

De manera que dicha resolución del sensor limitará la resolución que podremos configurar en nuestros lásers virtuales, el ángulo de visión o campo de visión limitará el rango de entorno (horizontal y vertical) que podremos visualizar sin necesidad de que el robot de mueva y por tanto también el rango de los lásers virtuales y por último en rango de profundidad limitará la medida máxima a la que se podrán detectar objetos sin necesidad de desplazar el robot, aunque puesto que este proyecto está orientado a entornos de interior no será una limitación importante.

Dicho sensor estará expuesto además a un cierto ruido, que se tratará en el capítulo 4 de este documento. Por tanto la información mostrada en el conjunto de láser virtuales vendrá limitada por las **especificaciones** del sensor (En la Figura

4.1 se pueden apreciar los sensores de Kinect y en la tabla 1 las características del sensor utilizado en las experiencias) y por la **configuración del usuario** que se explicará en el Anexo B.

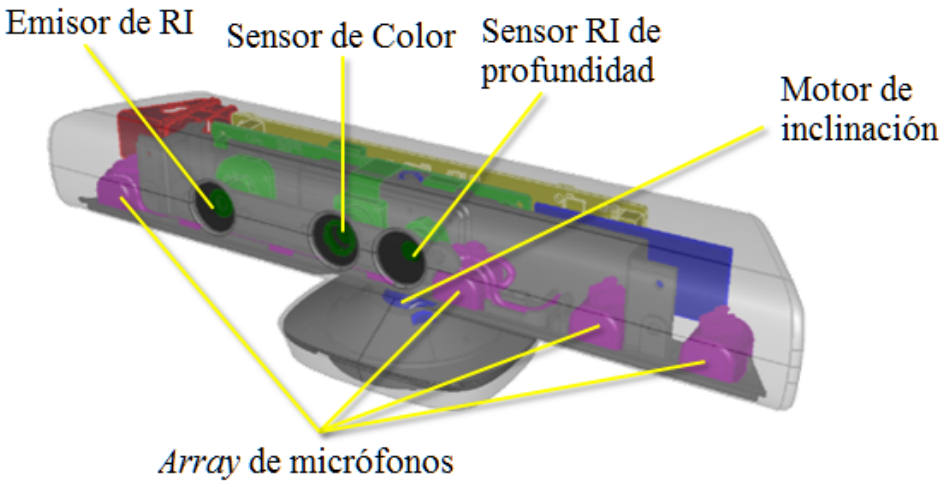

Figura 4.1: En la figura se pueden apreciar los diferentes componentes con los que está equipado el sensor Kinect utilizado en este proyeto, aunque no siempre se debe utilizar esta distribución. El emisor y receptor de infrarrojos para obtener la profundidad, el sensor que capta la información de color, los micrófonos incorporados y el motor de inclinación que permite mover la cámara dentro de un ángulo limite.

Cuadro 4.1: Parámetros del sensor Kinect utilizado

Parámetro	**Microsoft Kinect sensor**
Dimensión (mm)	14cm x 3.5cm x 5cm
Campo de visión (Horizontal, Vertical y Diagonal)	58º H, 45º V, 70º D
Resolución de profundidad	VGA (614x480)
Resolución espacial x/y (@ 2m de distancia del sensor)	3mm
Rango de operación	0.8m - 3.5m
Entorno de operación (Cualquier condición de iluminación)	Interiores

4.3. Generación de lásers virtuales

El primer paso en el proceso de descripción de entorno de este método que se presenta, sería la generación del conjunto de lásers virtuales a diferentes alturas, a partir de la información de profundidad obtenida por el sensor RGBD. Hay que tener en cuenta en primer lugar el **funcionamiento del algoritmo** que permite generar dicho conjunto de lásers y en segundo lugar habrá que tener en cuenta de que forma es **el error introducido** en las lecturas obtenidas finales.

4.3.1. Fases del algoritmo de generación

La generación del conjunto virtual de lásers se divide en 3 fases:

1. División de la *point cloud* almacenada por el sensor en subconjuntos de múltiples puntos separados por los planos paralelos al suelo y a ellos mismos.
2. Proyectar todos los puntos de todos los subconjuntos en el plano del suelo
3. Generar por último, las lecturas láser virtuales con cada uno de los subconjuntos de puntos

La separación y proyección de la *point cloud* es muy clara: Puesto que todos los planos son paralelos al suelo (en cada altura determinada), los puntos pueden ser particionados utilizando su altura en cada plano diferente. Es decir, se divide la nube de puntos asignando a cada punto su subconjunto, según los dos planos entre que se encuentre.
De forma similar, la proyección de los puntos de cada subconjunto se consigue ajustando su altura a cero. El algoritmo funciona de la siguiente manera:

1. Primero se generan N subconjuntos de puntos de los puntos totales tales $C_0...C_N$, donde cada C_i contiene todos los puntos entre los planos P_{i-1} y P_i. A dichos planos se le asigna una altura determinada en la fase de configuración (Se puede apreciar en la Figura 4.2).
2. En segundo lugar se genera el conjunto de *scans* de lásers virtuales $L_0...L_N$. Para conseguir dichos *scans*, primero las lecturas virtuales $p_i(j)$ son inicializadas con el valor máximo de distancia virtual del laser.

$$\forall i, j : i \in [0, N), j \in [0, M) \quad p_i(j) = \mathcal{M} \tag{4.1}$$

Donde M es el numero de lecturas por *scan* láser y $\mathcal{M}$ el máximo valor de distancia virtual del láser (como hemos dicho antes dependerá del RGBD y la posición del robot).

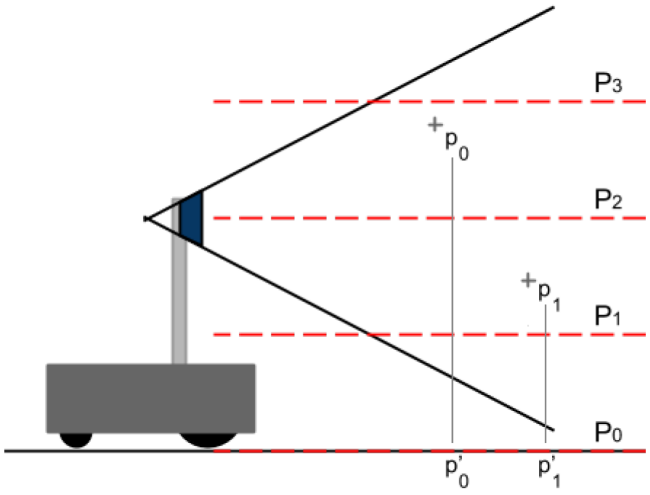

Figura 4.2: Demostración de como se puede dividir la nube de puntos capturada por el sensor RGBD en 4 planos de entradas (P_o, P_1, P_2 y P_3) y como se puede utilizar dichos planos para proyectar 2 lecturas de entrada (p_0, p_1). El punto p_0 correspondería al subconjunto de la nube de puntos C_2 y p_1 correspondería a C_1.

3. En tercer, y ultimo lugar, después de la inicialización para cada una de las lecturas de los lásers virtuales, se actualiza el valor de profundidad de los puntos que la componen. Cada punto $p_i(j)$ es convertido a coordenadas polares, de manera que a cada uno de ellos se le puede asignar una lectura específica del láser virtual. Por tanto, se aplica la siguiente condición: Si la distancia a un punto es menor que la distancia al punto de la lectura actual, el valor del primero es actualizado con el valor del de menor distancia, obteniendo siempre los valores más cercanos.

Al final en este proceso de generación del conjunto de lásers virtuales, la información proporcionada mediante un sensor láser virtual n en un scan simple se representa de la siguiente forma típicamente: $\{(r, \varphi)_l | l = 1...N_R\}_n$, donde $(r, \varphi)_l)$

son las coordenadas polares de la l-ésima lectura de distancia (se traduce como, r_l es el valor de distancia medido desde el sensor a alguno de los obstáculos del entorno del robot rotado en la dirección φ_l). En la Figura 4.3 se puede ver la interpretación de las lecturas para un telémetro láser, que será la misma que tengas los *scan* láser simulados.

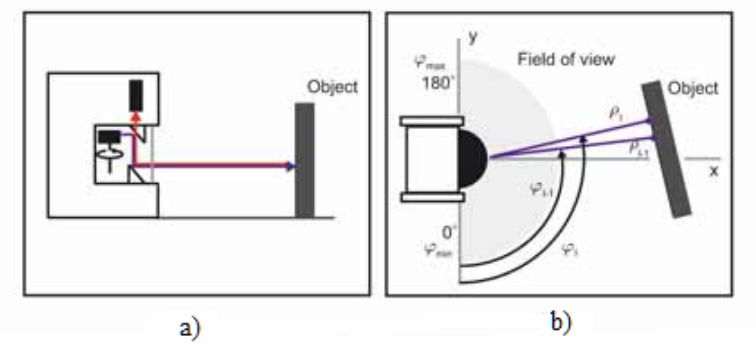

Figura 4.3: En la figura a) se puede apreciar la geometría que simula el generador de lecturas virtuales. En la figura b) se aprecian las referencias de cada una de las lecturas simuladas, idénticas a las de un telémetro laser.

De manera similar al telémetro láser real, las medidas de las lecturas virtuales son adquiridas mediante el sensor virtual con una resolución angular dada. En este caso esa resolución es $\triangle\varphi = \phi_l - \varphi_{l-1}$, la cual previamente será configurada por el usuario (Se ve bien ampliado en el Anexo B).

4.3.2. Error introducido

En todo este proceso de generación del conjunto de lecturas virtuales, hay que tener en cuenta el error introducido en las medidas, puesto que nuestro sensor no es un aparato ideal. Dicho error afectara principalmente a la distancia medida a cada punto de la lectura r_l. Se vera afectada por dos errores:

- Error sistemático (ϵ_s)
- Error estadístico (ϵ_r)
- Error por ruido "sal y pimienta": Más conocido en la literatura por su homónimo en inglés "*Salt and pepper*" Producido cuando el sensor no recibe señal de retorno

Por lo general se asume que estos errores siguen una **Distribución Gaussiana** con media cero y varianza σ_r^2. Estos errores no se tratan en la fase de generación del conjunto de lecturas virtuales puesto que se tratarán en la fase de **pre-procesado** perteneciente al algoritmo CUBA, y que utiliza ciertas técnicas para eliminarlo o minimizarlo, las cuales están ampliadas en [1].

4.4. Configuración del algoritmo de generación

Este proceso de generación de lásers virtuales surge como solución para poder realizar una descripción de entorno en 3D y compone una de las principales novedades de este proyecto. Es por tanto una fase muy importante a la hora de realizar la descripción de entorno a partir del algoritmo CUBA. Dicha generación debe ser adaptada y configurada según muchos aspectos como pueden ser: el entorno a describir, los objetivos del usuario, los resultados que se quieren obtener etc. Será punto de gran importancia la elección de una configuración que procuré una generación del conjunto de lásers virtuales satisfactoria para nuestros resultados finales. En conclusión, muchas configuraciones diferentes pueden darse dependiendo de las condiciones de aplicación de esté método de descripción, en donde se elegirán múltiples parámetros (Se amplia en el Anexo B de esté documento).

Capítulo 5

Descripción del entorno multi-capa mediante CUBA

Una vez vistos los objetivos e hipótesis planteados en el capítulo 3, y tras haber realizado una revisión de los diferentes enfoques más determinantes en la literatura referida a la descripción del entorno de un robot, en este capítulo vamos a analizar el sistema CUBA aplicado al conjunto de lecturas láser virtuales generadas que se explica en el capítulo 4. Como se explica en el capítulo 2 existen 3 tipos de clasificaciones para los métodos de representación de entorno, los cuales son: Cuantitativos, cualitativos e híbridos. Como ya se analizó, este método que se presenta está basado en un enfoqué basado en marcas naturales del entorno el cual utiliza información métrica del entorno para su representación, es decir un método cuantitativo. En definitiva, en éste capítulo se presenta el algoritmo CUBA aplicado a las múltiples capas en las que se divide el entorno y como funciona, de manera no muy extensa, pues en [1] se amplia detalladamente.

5.1. Introducción

El objetivo por tanto es la descripción del entorno en múltiples-capas como ya hemos visto, con la principal novedad: La información en esté método, es adquirida por el sensor RGBD y procesada para emular diferentes lecturas láser virtuales a diferentes alturas configuradas para obtener la descripción completa del entorno a mapear. Una vez generados el conjunto de lecturas láser, cada una

de ellas es analizada de acuerdo al algoritmo CUBA. Por lo tanto, el enfoque que propone este TFG obtendrá una descripción del entorno en 3D, donde los puntos de ruptura, bordes o extremos [13] y las 3 marcas de interés: **Segmentos rectos, esquinas (reales y virtuales) y segmentos curvos)** son detectados sobre cada una de las lecturas láser virtuales generadas. Una visión general sobre este enfoque viene ilustrada en la Figura 5.1. Los elementos que se detectarán al analizar la lectura láser se definen como sigue (En la Figura 2.3 se ve un ejemplo):

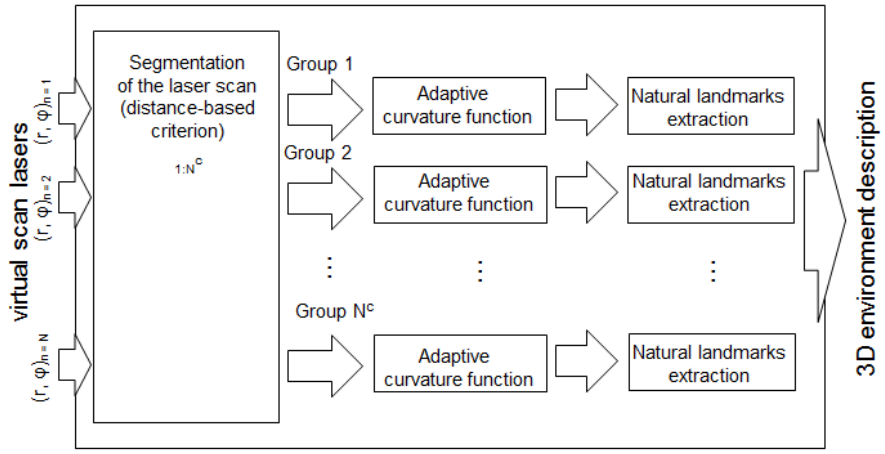

Figura 5.1: Esquema del conjuto de fases que sigue el algoritmo de descripcción de entorno propuesto en este proyecto

- **Puntos de ruptura** o *Breakpoints*: Son las mediciones sobre las lecturas láser virtuales asociadas a las discontinuidades producidas por cambios en la superficies escaneadas por el robot.
- **Segmentos rectos**: Son detectados al escanear una superficie plana (por ejemplo, una pared, una puerta cerrada, etc).
- **Esquinas reales**: Son detectadas al detectar un cambio en la orientación en la superficie escaneada por el robot.
- **Esquinas virtuales**: Son el resultado de dos segmentos previamente definidos.
- **Segmentos curvos**: Son elementos asociados a objetos circulares del entorno (por ejemplo, árboles, columnas, etc).

- **Bordes o extremos**: Corresponden con puntos de ruptura asociados a los extremos de la superficie escaneada por el robot (como pueden ser, los marcos de las puertas o esquinas convexas en las paredes).

Como ya hemos explicado a lo largo de este documento el sensor utilizado para adquirir la información métrica del entorno es un sensor RGBD (en la sección 4.2 del capítulo 4 se extendieron las características del sensor que se utiliza en este proyecto). El método consta de 3 etapas diferenciadas las cuales se resumen en las siguientes secciones.

5.2. Adquisición de datos y pre-procesado

En esta primera fase se incluye la obtención del conjunto de lecturas de exploración láser L_N obtenidas por el sensor RGBD, como ya se explica en el capítulo 4. Este conjunto de lecturas láser están afectadas por varios errores como se comento en el capítulo anterior, por tanto la primera fase del proceso será aplicar ciertas técnicas de corrección a la información obtenida por el sensor RGBD. La gama de lecturas láser se ve afectada por 3 errores diferenciados como ya mencionamos: El error sistemático, el error estadístico, y el error más conocido como ruido "sal y pimienta". La corrección aplicada se menciona a continuación resumidamente:

- El **error sistemático** es el introducido por el propio sensor y se manifiesta de igual manera en todas las medidas produciendo pequeñas variaciones en las distancias calculadas. Este error es reducido aplicando la aproximación polinomial descrita por Borges y Aldon [15].

- El segundo error es el conocido como **ruido "sal y pimienta"** el cual se debe a que el sensor de profundidad equipado en el RGBD no reciba señal de retorno (pimienta) o que se produzca un falso valor de profundidad debido a la alta reflexibidad de algunos materiales q (sal). Para poder solventar éste, se aplica un *filtrado de mediana* a las lecturas del sensor, que consigue reducir el número de puntos afectados por este error.

- Por último el **error estadístico** se produce al equipar el sensor RGBD en robot móvil, el cual se desplace libremente. Este error se corrige de una

manera intuitiva, considerando el movimiento del robot y compensando las medidas adquiridas por el sensor en función de la odometría en el instante inicial y final de la toma de datos.

Esta corrección del error produce unos resultados de las lecturas de distancia a partir de la información RGBD muy beneficiosos para la entrada a la siguiente fase del algoritmo, donde se tratarán más ampliamente.

5.3. Segmentación de la lectura láser virtual basada en información de curvatura

La primera fase de adquisición y pre-procesado obtiene por tanto un conjunto de lecturas láser virtuales a varias alturas sin error o al menos minimizado para que, el proceso de segmentación sea más eficaz. La segmentación por tanto consiste en dividir independientemente cada exploración láser en conjuntos de lecturas consecutivas dentro de la exploración, de acuerdo a un criterio de distancia. Siendo $(r, \varphi)_{i-1}$ y $(r, \varphi)_i$ dos lecturas consecutivas de una exploración, ellas pertenecerán al mismo segmento si y solo si la distancia entre ella es menor que un umbral dado. A diferencia de los métodos de segmentación clásicos, este algoritmo no utiliza un umbral de distancia fijo, sino que se adapta al entorno mejorando el proceso en situaciones concretas (pasillos, elementos lejanos, etc). Este método es utilizado por los autores en [15] para detectar los puntos de ruptura (*breakpoints*) de manera adaptativa, es decir, el algoritmo realiza la segmentación de cada una de las exploraciones a distintas alturas, en función de la distancia que separa dos puntos de manera adaptativa. En la Figura 5.2b se muestran los segmentos asociados a diferentes conjuntos o *clusters* para 2 exploraciones a diferentes alturas en el escenario dibujado en Figura 5.2a, dibujados en diferentes colores.

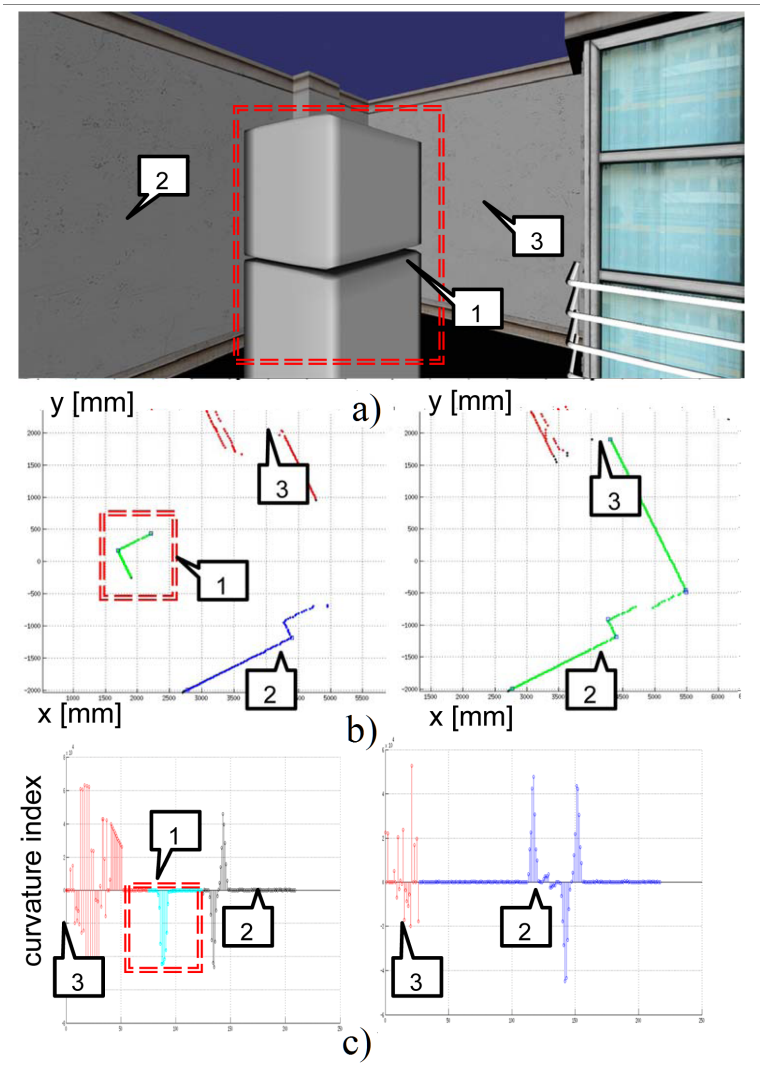

Figura 5.2: En la figura a) se muestra el entorno simulado con RCSimulator. En la b) los segmentos de los 2 *scan* laser virtuales (Con diferentes colores asociados a diferentes conjuntos); y en c) se muestran las funciones de curvatura asociadas a b) (Ver la sección 5.4)

Una vez agrupadas en *clusters* los diferentes *scan* a distintas alturas, la información de curvatura de manera adaptativa es utilizada para realizar una segmentación más profunda de cada uno, obteniendo los puntos asociados a lineas rectas y curvas. La función de curvatura describe básicamente cuanto se gira una curva en cada punto. Los picos de esta función de curvatura corresponden a las esquinas de la curva presentada, y su altura depende del ángulo de éstas. El índice de curvatura en cada lectura de la distancia del *scan* se filtra adaptativamente de acuerdo a la distancia entre las posibles esquinas en todo el *scan* láser (los K-puntos izquierda y los K-puntos derecha), permitiendo así que al no ser una

frecuencia fija no se eliminen esquinas y se elimine el ruido. Para cada lectura de la distancia i de un escaneo láser, se evalúan las coordenadas cartesianas. Para estimar la adaptación de curvatura en cada una de las lecturas, el método sigue los siguientes pasos:

- En primer lugar se calculan los valor $K_f(i)$ y $K_b(i)$, los cuales se definen como la máxima distancia del *scan* láser sin presentar discontinuidades a la derecha y a la izquierda de la lectura de profundidad i respectivamente. El valor de $K_f(i)$ se calcula comparando la distancia Euclídea entre i y su $K_f(i)$-ésimo prójimo ($d(i, i+K_f(i))$), con la distancia real de *scan* láser que existe entre ambas lecturas de profundidad ($l(i, i+K_f(i))$). Estas distancias tienden a ser la misma en ausencia de esquinas, incluso si los *scan* láser son ruidosos. Por el contrario, si existen esquinas, la distancia Euclídea es considerablemente más corta que la distancia real. Por tanto $K_f(i)$ sería el valor más grande que satisface:

$$d(i, i + K_f(i)) > l(i, i + K_f(i)) - U_k \tag{5.1}$$

  donde U_k es un valor constante que depende del nivel de ruido tolerado por el detector. $K_b(i)$ es también asignado de acuerdo a la Eq. 5.1, pero utilizando $i - K_b(i)$ en lugar de $i + K_f(i)$. En presente proyecto, ha sido experimentalmente demostrado que U_k es igual a 1.0 trabajando correctamente.

- En segundo lugar, se calculan los vectores locales $\overrightarrow{f_i}$ y $\overrightarrow{b_i}$ asociados a cada lectura de profundidad i. Estos vectores representan la variación en el eje x e y entre las lecturas i y $i + K_f(i)$, y entre i y $i - K_b(i)$. Si (x_i, y_i) son las coordenadas de la medidas de profundidad i, el vector loca asociado es definido como:

$$\begin{aligned} \overrightarrow{f_i} &= (x_{i+K_f(i)} - x_i, y_{i+K_f(i)} - y_i) = (f_{x_i}, f_{y_i}) \\ \overrightarrow{b_i} &= (x_{i-K_b(i)} - x_i, y_{i-K_b(i)} - y_i) = (b_{x_i}, b_{y_i}) \end{aligned} \tag{5.2}$$

- El tercer paso consistiría en el calculo del índice de curvatura $|K_\theta(i)|$, el cual representa enl angulo asociado a la medida i. Este ángulo puede ser estimado de la siguiente forma, para una medida i:

$$|K_\theta(i)| = arccos\left(\frac{\overrightarrow{f_i} \cdot \overrightarrow{b_i}}{|\overrightarrow{f_i}| \cdot |\overrightarrow{b_i}|}\right) \tag{5.3}$$

- En último lugar, se detectan las esquinas sobre $|K_\theta(i)|$. El índice de curvatura obtenido representa la curvatura asociada a cada medida en valor absoluto. Las medidas identificadas como esquinas satisfarán las siguientes condiciones: *i*) Son picos locales de la función de curvatura y *ii*) sus $|K_\theta(i)|$ valores superan el ángulo mínimo requerido para ser consideradas esquinas, en lugar de falsos picos debido al ruido restante (θ_{min})

La figura 5.2 c) muestra las funciones de curvatura asociadas a los *scan* láser virtuales mostrados en la Figura 5.2 b.

5.4. Extracción de marcas naturales a partir de la función de curvatura

El segmentador en dos pasos (Detección de los *breakpoints* y organización en *clusters* y segmentación por curvatura) descrito en la fase anterior, presenta a su salida un conjunto de puntos asociados a segmentos rectos, curvos y esquinas en el entorno del robot. Esta será la ultima fase del algoritmo CUBA, en la cual a partir de los segmentos se obtienen el conjunto de marcas naturales para cada *scan* láser, describiendo el entorno del robot. Para extraer dichas marcas se calcula la localización de cada marca en el espacio a partir del ajuste de los puntos a modelos matemáticos, así como la incertidumbre asociada a esta posición.

La función adaptativa de curvatura $|K_\theta(i)|$ puede directamente proporcionar 3 marcas naturales diferentes. A fin de incluir estos elementos como marcas en futuras tareas de robótica, es necesario caracterizar dichas marcas. A continuación, $|K_\theta(i)|$ se analiza para extraer el conjunto de características del entorno del robot, que se consigue mediante el ajuste de curvas paramétricas a las medidas asociadas

a cada segmento recto o curvo. Cada tipo de marca se caracteriza de la siguiente forma:

- Detección de los segmentos rectos a partir de $|\kappa_\theta(i)|$. Los segmentos rectos son el resultado de la exploración de superficies planas. Por lo tanto, son aquellos conjuntos de medidas consecutivas los cuales: i) están por debajo del ángulo mínimo de curvatura (En los resultados experimentales propuestos, esta altura mínima de curvatura, θ_{min}) ha sido fijada a 0.05); y ii) tienen un tamaño superior a un valor mínimo de longitud ($l_{min} = 10$ medidas). Con el fin de caracterizar los segmentos rectos, se usa el siguiente método descrito en [1]. Esté método utiliza una regresión lineal que aproxima el conjunto de puntos de una linea recta a la ecuación punto-pendiente (α, el angulo entre el eje x y la normal de la recta, y d la distancia perpendicular de la recta a su origen).

- Detección de los segmentos curvos sobre $\kappa_\theta(i)$. Los segmentos curvos son el resultado de la exploración de superficies curvas. Por el contrario que los valores de curvatura asociados a los segmentos rectos, se puede apreciar que la función de curvatura asociada a los segmentos curvos presenta un conjunto de picos locales consecutivos, pudiendo ser considerados erróneamente algunos de ellos como esquinas. Para evitar este error, el algoritmo está asociado a un índice de esquina para cada conjunto de medidas consecutivas cuyos valores $\kappa_\theta(i)$ están sobre θ_{min} o por debajo de $-\theta_{min}$ y tienen un tamaño superior a l_{min} . Este índice de esquina, ci, se define como:

$$ci = \frac{\frac{1}{i_e - i_b} \sum_{j=i_b}^{i_e} \kappa_\theta[j]}{max_{i \in (i_b, i_e)} \{\kappa_\theta[i]\}} \tag{5.4}$$

  donde i_b y i_e son las medidas que limitan el posible segmento curvo. Si ci es cercano a uno, la curvatura media del segmento y el valor máximo son parecidos, y se puede considerar un segmento curvo. Si ci es pequeño, la media de curvatura del segmento es inferior que el valor máximo, luego entonces, el segmento no puede ser considerado como segmento curvo. Por lo tanto, los segmentos curvos son aquellos conjuntos de medidas consecutivas las cuales no definen un segmento recto y tienen un indice de esquina más

grande que un umbral dado U_c (U_c ha sido fijado a 0.5 en todos las pruebas). Finalmente, el conjunto de puntos es entonces fijado como un circulo (es decir, centrado en coordenadas (x_0, y_0) y radio ρ).

- La detección de esquinas sobre $\kappa_\theta[i]$. Las esquinas son fácilmente detectables a partir del análisis de la función de curvatura como un valor asociado a un pico local, y a una región limitada por dos medidas, i_b y i_e. Por tanto, $\kappa_\theta[i]$ debe tener un valor sobre el mínimo angulo requerido para ser considerado una esquina en lugar de un falso pico debido al ruido restante. Una vez que una esquina real es detectada, su posición (x_c, y_c) es estimada como la intersección de las dos rectas que la generan. Las esquinas virtuales son definidas como la intersección de segmentos rectos prolongados las cuales no son previamente definidas como esquinas reales. Por lo tanto, las esquinas virtuales también son caracterizadas de la misma forma que las reales.

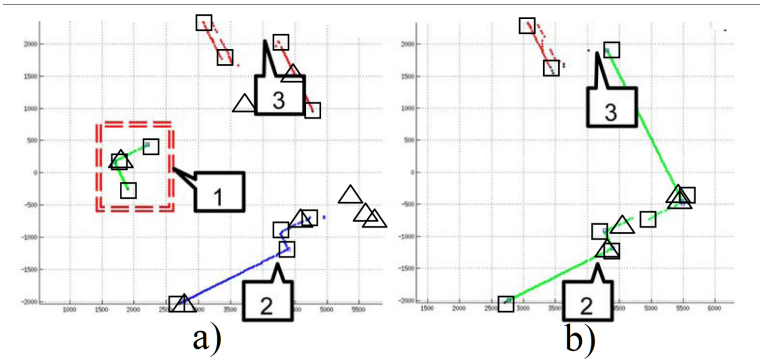

Figura 5.3: En la figura a-b se muestra la detección de las marcas naturales asociadas a la Figura 5.2 (cuadrados - segmentos rectos y puntos de rupturas, triangulos - esquinas)

La figura 5.3 muestra las marcas naturales extraidas por el método propuesto a partir de los *scans* láser virtuales presentados en la figura 5.2b.

Capítulo 6

Resultados Experimentales

En este capítulo se presentan las experiencias realizadas con el fin de evaluar el rendimiento de este nuevo método presentado. Las pruebas se realizarán tanto en entornos del robot simulados mediante software, como en entornos reales. Como ya hemos explicado en capítulos anteriores, el método propuesto es capaz de extraer hasta 5 marcas naturales del entorno, a partir de la información que aporta el sensor RGBD, para cada uno de los niveles de altura configurados. Los algoritmos implementados que engloban este proyecto han sido desarrollados en en lenguaje de programación *C++* y los test de prueba han sido realizados en un PC con procesador Intel Core i5 2.4Ghz con 4Gb de DDR3 RAM y GNU-Linux Ubuntu 13.10. Las características del sensor RGBD utilizado se pueden ver en la figura 4.1. Los experimentos han sido enfocados en la evaluación del algoritmo en términos de robustez, numero de características detectadas y recursos computacionales. Además se realizará un estudio comparativo del método propuesto con los algoritmos de descripción de entorno previos presentados en [1].

6.1. Arquitectura software

El software para controlar el sistema propuesto se basa en el *Framework* de robótica RoboComp [14], al cual se le dedica el Anexo A de esté proyecto, donde se explicarán sus principales funciones. Como se explicará en dicho anexo, este *Framework* utiliza programación orientada a componentes que se interconectan entre sí. En este nuevo método de descripción de entorno las principales noveda-

des eran la utilización del sensor RGBD y la descripción del entorno en 3D. Para la generación del conjunto de lásers virtuales explicada en el Capítulo 4, se utiliza el nuevo componente implementado *laserRGBD*, el cual divide la información RGBD (*rgbdcomp*) en varios *scan* láser funcionando en paralelo.

A la salida de éste componente se aplicaría el algoritmo CUBA para la extracción de las marcas naturales de cada plano. El componente *cubafeaturesComp* una vez ha caracterizado el entorno, ofrece a su salida una estructura que almacena los parámetros necesarios de todas estás características Por último se ha creado un nuevo componente, *curvature3Dcomp* el cual almacena todas las características extraídas del entorno en una misma estructura de datos para que pueda ser procesada en futuros trabajos. En la figura 6.1 se muestrán todos los componentes utilizados en este método presentado.

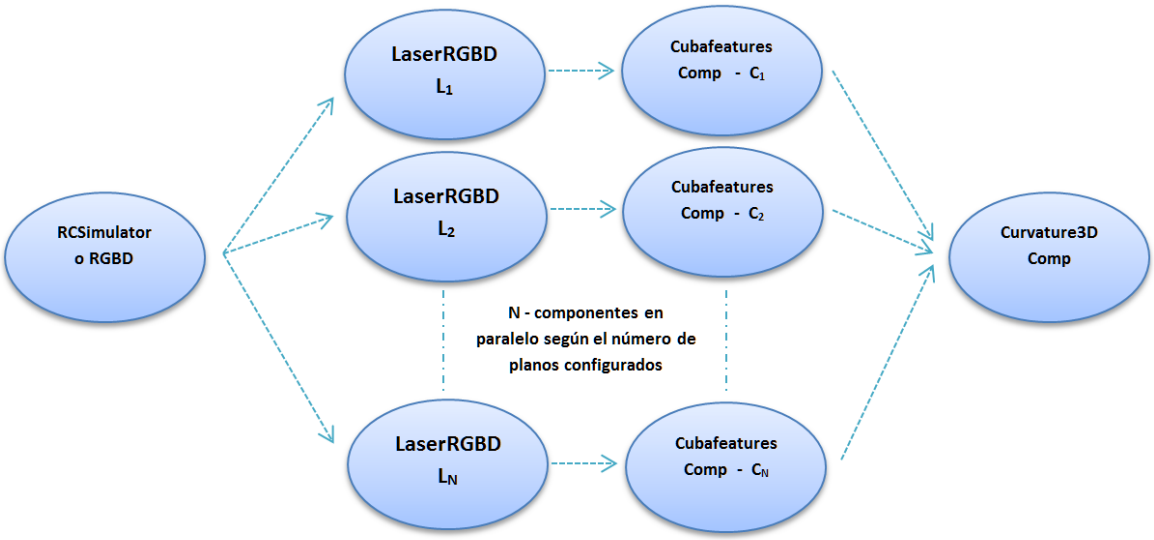

Figura 6.1: Esquema de como heredarían e interactuarían los diferentes componentes que utiliza este método en la descripción del entorno, implementados en RoboComp

6.2. Entornos simulados

Para poder hacer pruebas sin necesidad de un despliegue real del robot con sensores, se utiliza una herramienta que pertenece a RoboComp llamada RoboComp InnerModel Simulator (RCIS), el cual nos permite obtener los resultados simulados. RCIS es un simulador de robot en 3D diseñador para uso en el ámbito

académico, en etapas tempranas de desarrollo, y principalmente, para propósitos de investigación. Una de sus características más notables es que permite a los usuarios controlar el ruido producido por los sensores y actuadores simulados. Esta propiedad se puede utilizar para probar cómo de robustos son los algoritmos frente el ruido y, si el ruido se configura nulo, para diferenciar entre los problemas relacionados con errores de ruido y errores de algoritmos. La herramienta RCIS se verá ampliada en el Anexo A.

Para las pruebas en entornos simulados de este proyecto se simulan diferentes sensores RGBD con diferentes niveles de ruido en un escenario de interior. El uso de este simulador nos permite realizar una evaluación de la robustez de los algoritmos de acuerdo a diferentes valores de ruido del sensor. La figura 6.2a muestra el entorno simulado usado en los experimentos. El escenario simulado para las pruebas, está compuesto por una habitación simple, un cilindro y dos cajas de diferentes tamaños (2 y 3 en la figura 6.1a). En la figura 6.2b se muestra la segmentación del entorno del robot utilizando 5 *scan* láser virtuales. Por ultimo, las marcas naturales son detectadas mediante el algoritmo para los *scans* láser virtuales L_1 y L_3 son ilustradas en la figura 6.2c. Los puntos límites de los segmentos rectos, las esquinas y los segmentos curvos, son ilustrados como cuadrados, triángulos y circunferencias, respectivamente.

El escenario simulado ha sido utilizado para evaluar el número de marcas naturales detectadas por el algoritmo propuesto, y como depende del error por ruido añadido al sensor RGBD y al número de *scans* láser virtuales. Dichos resultados se resumen en la Tabla 1 y la Tabla 2. Como muestra la Tabla 1, la carga computacional permite trabajar en tiempo real (es decir, todos los componentes en paralelo), lo cual produce un aumento considerable del número de marcas detectadas (n_c, n_{ls}, n_{cs} y n_t son el número de esquinas, segmentos rectos, segmentos curvos y número total de marcas naturales detectadas por el algoritmo respectivamente).

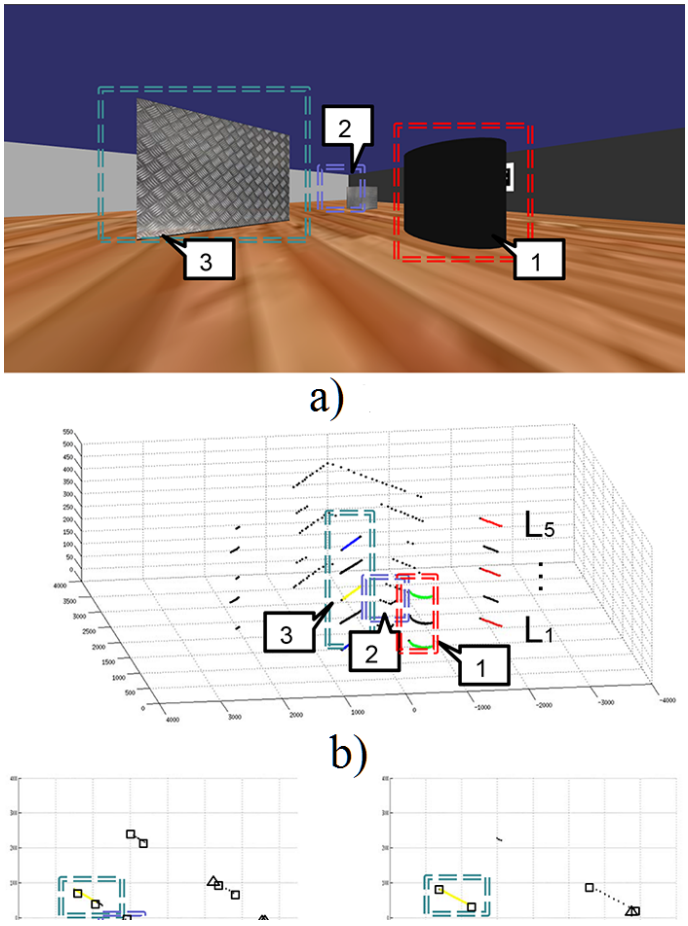

Figura 6.2: La figura a) muestra el escenario simulado utilizado en el experimento; la figura b) el cojunto de *scan* láser virtuales ($L_i|i = 1..,5$); y la c) muestra las marcas naturales extraidas por el método para 2 diferentes *scan* láser, L_1 y L_3

Cuadro 6.1: Resultados experimentales del algoritmo para entorno simulado

Tiempo consumido (ms)	n_c	n_{ls}	n_{cs}	n_t	L_i
6.0	5	7	1	13	L_1
6.0	5	7	1	13	L_2
5.9	2	3	1	6	L_3
5.9	2	3	0	5	L_4
5.9	1	3	0	4	L_5
5,95	**15**	**23**	**3**	**41**	$\mathbf{L_{N=5}}$

El siguiente método de evaluación definirá la robustez del algoritmo de descripción de entorno según 2 métricas diferentes como ya se realizó en [1], comparando

los resultados con enfoques similares y obteniendo mejores resultados. Para esté estudio comparativo, el enfoque presentado se comprará únicamente con el anterior enfoque. Las dos métricas que se van a utilizar se definen como:

$$TruePos = \frac{NumberMatches}{NumberTrueSeg} \tag{6.1}$$

$$FalsePos = \frac{NumberSegExAlg - NumberMatches}{NumberSegExAlg} \tag{6.2}$$

Donde $NumberSegExAlg$ es el número de segmentos extraídos por el algoritmo, $NumberMatches$ es el número de coincidencias de segmentos verdaderos y $NumberTrueSeg$ es el número de segmentos verdaderos. Puesto que como ya hemos dicho, el estudio se realizará frente proyecto anterior, el cual únicamente describía el entorno a una altura concreta, hemos tomado únicamente una de las alturas del conjunto de lecturas (L_3) y se han añadido diferentes valores de ruido a las información RGBD.

Cuadro 6.2: Estudio comparativo con el trabajo previo de este proyecto (ver [1])

Núñez et al. ([1])	Propuesto ($L_{N=5}$)	Algoritmo
0.875	0.825	TruePos (σ=0m)
0.0	0.03	FalsePos (σ=0m)
0.815	0.79	TruePos (σ=0.01m)
0.02	0.12	FalsePos (σ=0.01m)
0.750	0.70	TruePos (σ=0.02m)
0.8	0.18	FalsePos (σ=0.02m)

Para este experimento se han utilizado 3 valores diferentes de ruido añadidos al sensor RGBD. La distancia de ruido se ha simulado utilizando una distribución normal con $\mu = 0$, y con varianza 0, 0.01 y 0.02 metros, respectivamente. En la Tabla 2 actualiza los resultados descritos en [1], incluyendo los resultados obtenidos en el enfoque que se propone. Como conclusión, la robustez de el método propuesto es muy similar al anterior, teniendo en cuenta diferencias insignificantes.

6.3. Entornos reales

Para las experiencias en entorno real se utilizo el robot RobEx, el cual adquirió los datos en el entorno utilizado para las pruebas (Figura 1.1). El robot RobEx es un robot diferencial diseñado por RoboLab en la Universidad de Extremadura. Para las experiencias realizadas en este apartado, se utilizo un sensor RGBD montado en la parte superior del robot y se configuro para adquirir la información de profundidad del entorno a 30 fps. A partir de los facilidades de RoboLab el robot fue teloperado al área de test. Para crear un entorno con suficientes características

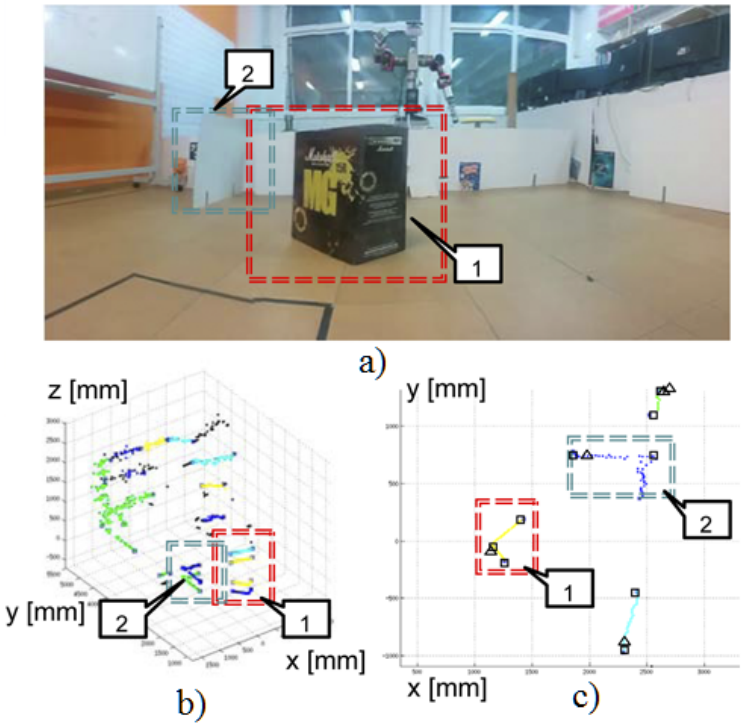

Figura 6.3: En la figura a) se muestra el escenario real utilizado en los experimentos; y en la figura b) se muestran las marcas naturales extraidas mediante el método propuesto utilizando 5 *scans* láser vituales a diferentes alturas.

para caracterizar, se utilizaron cajas y falsos muros situados en frente del robot. La Figura 6.3a muestra el área utilizada para las pruebas. En la Figura 6.3b, se muestran gráficamente las 5 lecturas virtuales láser que se han utilizado para la prueba. En la figura 6.3c se muestra un ejemplo de la extracción y caracterización de marcas a partir de la lectura L_3. Para finalizar las pruebas en la Tabla 3 se muestran los resultados de la aplicación del algoritmo en este escenario, donde

se puede apreciar que el número de marcas ha aumentado considerablemente respecto del uso de una única lectura.

Cuadro 6.3: Resultados del algoritmo en entornos reales ($L_{N=5}$)

n_c	n_{ls}	n_{cs}	n_t
27	20	0	**47**

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

En éste capítulo se presentan las principales conclusiones de es TFG y el trabajo futuro a partir de este. Se ha presentado una descripción multi-capa del entorno de alrededor del robot mediante un conjunto de *scans* láser virtuales que se configuran a diferentes alturas mediante el algoritmo de división de la nube de puntos de profundidad, obtenidos a partir de un sensor RGBD de bajo coste. Estos *scans* láser virtuales son procesados para extraer las marcas naturales del entorno en base a su información de curvatura en tiempo real, en los cuales se emplean las 3 fases ya explicadas: Adquisición de datos y pre-procesado, segmentación y por último la extracción de características. Este enfoque extiende el trabajo previo de los autores a escenarios 3D, aumentando el número de marcas detectadas y caracterizadas mediante el algoritmo, dando pie a un buen número de aplicaciones futuras (En la sección de Trabajo Futuro se explicarán algunas). El método presentado ha sido evaluado de manera eficaz en escenarios reales y simulados, y comparado con enfoques previos, donde se ha podido comprobar la robustez de este algoritmo, su eficiencia en la función de extracción de marcas del entorno obteniendo una descripción satisfactoria y sin necesidad de una gran carga computacional. Todo el algoritmo y sus diferentes partes se han implementado mediante RoboComp, el cual es un *Framework* muy eficiente y con gran funcionalidad, y por tanto todo este proyecto formará parte del mismo. Se puede concluir en este apartado comentando que la hipótesis principal ha sido imple-

mentada satisfactoriamente y cubriendo todos los objetivos, competencias tanto principales como transversales.

7.1.1. Contribuciones

Con este método de descripción de entorno propuesto en este TFG se ha contribuido a la realización de un artículo para el XV Workshop of Physical Agents (WAF) [16]. Dicho artículo compone un resumen de esté algoritmo de descripción de entorno que se presenta. Además también se ha contribuido al añadirse los componentes generados al repositorio de RoboComp, permitiendo poder utilizarse este trabajo para aplicaciones futuras como las explicadas en la siguiente sección.

7.2. Trabajo futuro

Todo éste método de descripción de entorno es parte de un anterior proyecto de futuro y da pie a muchas posibilidades. En el punto final de esté método obtenemos un entorno de interior descrito por múltiples marcas naturales del entorno a diferentes niveles, definiendo cada plano. Por tanto todo el trabajo futuro será enfocado al uso de estás marcas naturales para tareas robóticas más complejas, de las cuales vamos a enumerar algunas:

- **Nuevas marcas del entorno:** Una vez se ha caracterizado el entorno en diversas marcas diferenciadas en distintos niveles, uno de los posibles pasos siguientes sería la creación de nuevas marcas a partir de estás, las cuales permitan caracterizar el entorno de una manera global. Es decir, una vez hemos obtenidos los segmentos rectos, curvos y las esquinas, se pueden definir nuevas marcas que ocupen varios planos como unión de las anteriores, ya sean puertas, paredes, ventanas, columnas, objetos cúbicos etc.

- **Aumento del número de marcas:** En el anterior trabajo [1] se realiza un ensayo de aplicación de SLAM a dicho método. La presencia de marcas únicas en los ambientes y su detección y extracción fiable es esencial para la aplicación del SLAM. Esto nos lleva a requerir una detección de más

marcas, que procuraría una mayor del SLAM en ciertos entornos. Esto se podría realizar mediante el uso de conceptos de alto nivel y conocimiento previo acerca de los ambientes típicos (por ejemplo, ventanas abiertas, archivadores, etc).

- **SLAM y Localización:** EL siguiente paso para cualquier algoritmo de navegación sería la navegación, propiamente. El método presentado conjunto a un algoritmo de trazado de rutas en tiempo real permitirá al robot desplazarse a la vez que describe el entorno. Puesto que éste método tiene una carga computacional moderada será bastante eficaz para este tipo de aplicaciones

- **Información de color añadida:** La principal novedad de este método como ya se ha explicado a lo largo de este documento, consiste en la utilización de un sensor RGBD para obtener la información en bruto que permitirá caracterizar el entorno. Dicho sensor obtiene información de profundidad y de color, por tanto una mejora importante para este método consistiría en añadir información de color (RGB) a la etapa de segmentación, realizando una segmentación más precisa y eficaz. Además añadir dicha información de color contribuiría además a una mejor definición de marcas del entorno, y sería bastante positivo para las mejora de nuevas marcas que se ha comentado en el primer punto de esta sección.

Estas son algunas de los principales enfoques para trabajo futuro, aunque este método define muchas nuevas posibilidades gracias a su versatilidad.

7.3. Evaluación Personal

Para finalizar está memoria de trabajo de fin de grado, comentar mis conclusiones personales y mi experiencia a lo largo de la elaboración de este nuevo método de descripción de entorno. El punto de partida de este proyecto es un el método descrito en [1], del cual he aprendido mucho a lo largo de la elaboración de está mejora. A todo el aprendizaje del método anterior hay que añadirle la familiarización con RoboComp, además del aprendizaje de los componentes utilizados en

este método y la profundización en varios lenguajes de programación. Este conjunto de elementos conlleva una curva de aprendizaje complicada al inicio de éste proyecto y que a medida que se avanza, se suaviza considerablemente. Considero que éste proyecto ha contribuido mi propio aprendizaje sobre bastantes conocimientos tanto de robótica, como de programación orientada a componentes, *C++*, XML, elaboración de proyectos etc. Este método desarrollado puede contribuir a bastantes aplicaciones futuras como ya se ha mencionado, en un gran abanico de campos, traduciéndose en una experiencia muy productiva y gratificante.

Apéndice A. Robocomp: Framework orientado a componentes

En este anexo se comentará de manera breve, en que consiste el **RoboComp** [14], el *framework* donde se basa todo este proyecto y en el cual se ha implementado. Estará orientado a las funciones que utiliza el método presentado en este proyecto.

A.1. Introducción y principales características

Este *framework* fue creado con el objetivo de facilitar a la comunidad robótica una manera de abordar la complejidad código, la escalabilidad o reutilización de código, etc. Estos problemas específicos están a la orden del día y por eso se toman iniciativas basadas en Components Oriented Programming (COP), aunque muchos más enfoques se han dado. RoboComp es por tanto un *framework* de robótica de *open-source* que se centra en la facilidad de uso. Esta basado en **Ice** [18] un middleware de calidad industrial, aunque su principal punto fuerte es el conjunto de herramientas que se acompañan, facilitando la programación. En la web [17] además se puede encontrar una amplia documentación en linea para la iniciación en el *framework*.

Al estar basado en **Ice**, reúne todos los requerimientos básicos para un *framework* de robótica, como son: Múltiple suporte de lenguajes (C++,C#, Java, etc), múltiple suporte de plataformas (Linux, Windows, Mac OS X, etc), comunicación eficiente, etc. Esto permite conseguir un *framework* completo en donde los

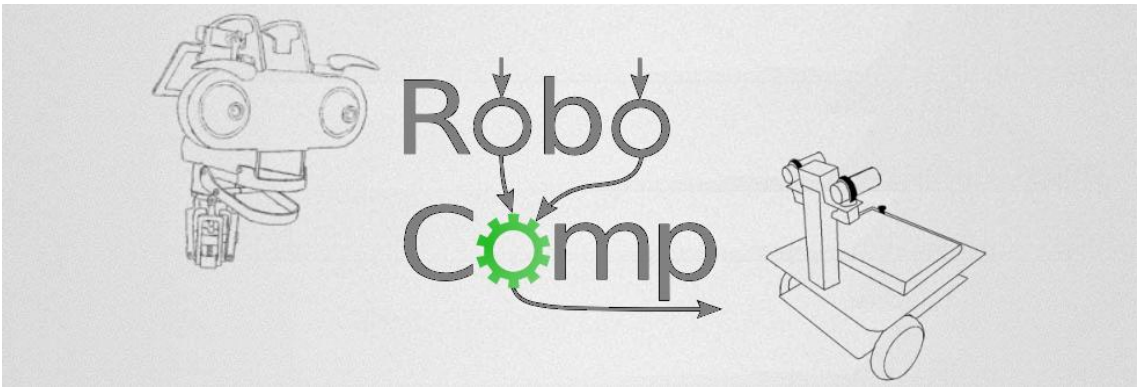

Figura A.1: Logo de RoboComp

componentes pueden ser integrados fácilmente con los existentes, e interconectados. Que este software esté basado en Ice además proviene a RoboComp de un numeroso número de herramientas y clases con las que está equipado, facilitando desarrollar sistemas complejos. Una de las clases más importantes que debemos resaltar se denomina *InnerModel*, la cual consiste en la propiocepción del robot, definido en XML y en la que se basa cualquier sistema robótico.

Por último, una de las contribuciones más importantes la cual RoboComp apoya firmemente: Hadware Abstraction Layer (HAL). Es la base para la reutilización de código, escalabilidad y la portabilidad. Todos los sensores y actuadores del robot actuarán sobre una misma interfaz lo que contribuirá a: i) Diferentes usuarios, investigadores o compañías pueden compartir sus componentes independientemente del hardware suyaciente, ii) se reduce el impacto de actualización de hardware, iii) previene la desaprobación de software. Algunas de as interfaces definidas en RoboComp son: Camera, DiferentialRobot, GPS, Láser, etc.

A.2. Component Oriented Programming

Para poder conseguir un fácil desarrollo, una buena estructura de componentes debe ser definida. El esqueleto de los componentes de RoboComp (Figura A.2) se compone de 3 principales elementos: La interfaz de servidor, el *worker* y los proxys para la comunicación de otro componentes. La clase *worker* será la responsable de implementar el núcleo de funcionalidad de cada componente La interfaz de servidor es la clase derivada a partir del Ice IDL, la cual implementa

los servicios que ofrece cada componente, la cual interactuará principalmente con el *worker*. Los proxys serán también parte importante, obtenidos normalmente por el *worker*. Son instancias de clases auto generadas que constituyen el acceso a otros componentes Además los componentes incluyen un archivo de configuración donde todos los parámetros de operación y comunicación son especificados. Utilizando dicha configuración, el programa principal generará la inicialización necesaria para ejecutar el componente.

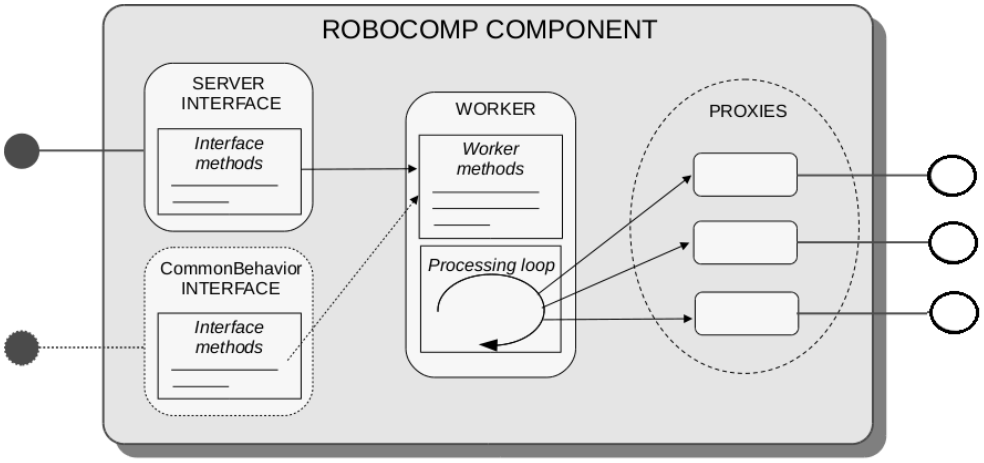

Figura A.2: Esqueleto de componentes de RoboComp

A.3. Herramientas

Para finalizar este capítulo, vamos a mencionar las dos herramientas mas importantes para el desarrollo de este proyecto. Estas herramientas consisten en características que complementarán la capa de *Middelware*:

- **componentGenerator:** Esta herramienta permite que el proceso de generación de nuevo componente sea automatizado. Puesto que RoboComp utiliza una estructura DSL, el nombre de esta herramientas será DSLEditor, liberando al programador de problemas de *MiddelWare*. En la web de RoboComp, existen varios tutoriales para la creación e interconexión de componentes [17].

- **Soporte de simulador:** También denominado **RCSimulator**. Esta herramienta ha permitido realizar *debugging* sobre el software desarrollado sin

necesidad de tener un robot activo, además de permitir realizar varias experiencias variadas. RoboComp permite además utilizar simulación en 2D o en 3D y se incluye en la sección de herramientas HAL que se ha comentado anteriormente. Con la colaboración de otros componentes HAL, como es el componente **joystick** permiten realizar funciones con mucha más facilidad. En la figura A.3 se puede apreciar una imagen de un entorno simulado de interior con dicha herramienta.

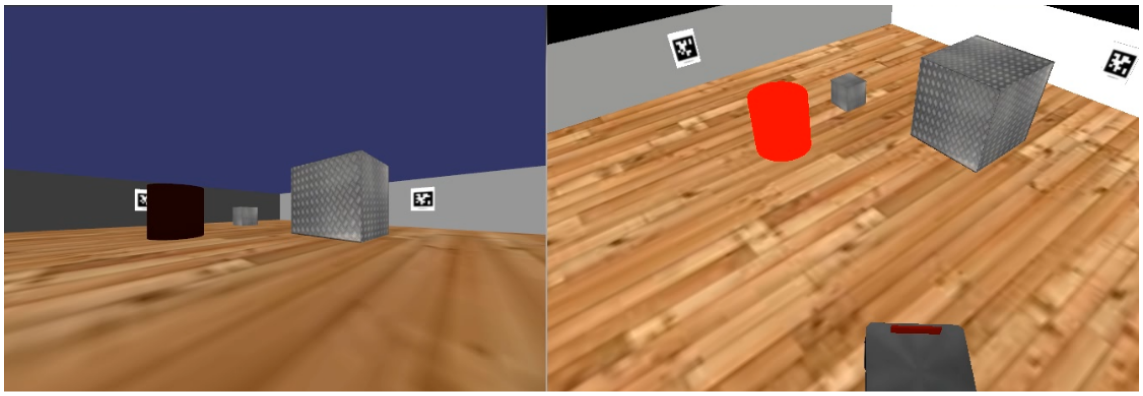

Figura A.3: En la figura a) se aprecia un entorno simulado visto desde el sensor; en la b) se aprecia el entorno visto desde el exterior del robot

Apéndice B. Parámetros de configuración en la descripción de entorno

En éste apéndice como se ha indicado a lo largo de este documento se va a explicar la valoración de ciertos parámetros que son configurados a la hora de utilizar éste método de descripción de entorno. Como se ha visto en el apéndice anterior, cada uno de los diferentes componentes que pueden ser implementados dentro de RoboComp, tiene asignado un fichero de configuración que automáticamente genera el *framework*. Dicho fichero contiene las información necesaria para interconectar los componentes que son ejecutados en tiempo real (por ejemplo, se definen las dirección de los equipos en donde se están ejecutando un componente necesario), añadir valores variables externos al propio componente, etc. En concreto para este método de descripción de entorno los parámetros de configuración que vamos a explicar estarán referidos al componente *laserRGBD*, el cual genera el conjunto de lecturas láser virtuales a partir de la información que le transmitirá el sensor RGBD, el componente *cubafeaturesComp* que consiste en el algoritmo CUBA propiamente y por último al componente *Curvature3Dcomp* que consistirá en la agrupación de todas las marcas naturales extraídas del entorno a todos los niveles de altura configurados.

B.1. Parámetros LaserRGBD

El archivo de configuración del componente *laserRGBD* podremos definirlo con el nombre que queramos pero con la extensión para configuración, tal que: *xxxxx.conf*.

A la hora de ejecutar el componente tendremos que tener en cuenta el nombre del archivo de configuración. Dentro de dicho archivo se podrán configurar los siguientes parámetros:

- **laserRGBD.Endpoints:** Se definirá el puerto en donde se ejecutará en tiempo real el componente laserRGBD (Donde se enviarán los datos de salida de dicho componente).
- **DifferentialRobotProxy:** Se definirá el puerto en donde recibe la información del robot utilizado.
- **InnerModelPath:** A este parámetro se le asignará la ruta en donde está almacenado el archivo XML que definirá el entorno en el que se está trabajando. Si es un entorno simulado el archivo XML tendrá toda la información sobre la posición de todos los objetos del entorno, el robot, los sensores con los que está equipado, los puertos utilizados etc. Si el mundo es real, únicamente definirá la posición de sensores y robots y sus puertos.
- **Parámetros del sensor RGBD:** Para este apartado habrá varios parámetros:
  - RGBDNumber: Definirá el número de cámaras utilizadas, ya que se podrá acceder a varias.
  - LaserBaseID: Definirá el nombre identificativo con el que se ha denominado a la base del robot en el fichero XML para entornos simulados o en el componente que gestiona el sensor para entornos reales.
  - MinHeight y MaxHeight: Es el parámetro más importante a la hora de la configuración del conjunto de *scans* virtuales. Para cada plano explorado del entorno habrá que utilizar un componente *laserRGBD* que será descrito a continuación por el algoritmo CUBA. Con estos dos parámetros se definirá el rango de altura que cubrirá esté nivel de exploración laser tal que: L_x tomará muestras de profundidad para $[MinHeight, MaxHeight]$.
  - LaserSize: Definirá el número de lecturas individuales para cada *scan* virtual láser (Equivalente a la resolución del telémetro láser).

- MinRange y MaxRange: Definirá los valores de profundidad mínima y máxima a las que se acotará el *scan* láser generado a partir de la información RGBD.
- FOV: Definirá el valor de distancia focal que se utilizará para interpretar los datos obtenidos por el sensor.
- RGBDProxyX: Este parámetros definirá el puerto para recibir la información del sensor RGBD. El valor X se modificará según el valor asignado al proxy del sensor, para diferenciarlos cuando haya más de un sensor funcionando a la vez.
- RGBDIDX: Se le asignará el nombre identificativo del proxy del sensor X, que vendrá ligado al anterior parámetro.

B.2. Parámetros CubafeaturesComp

Al igual que para el archivo de configuración del componente *LaserRGBD* la extensión será .conf y el nombre el que se asigne. Los parámetros de configuración en este caso serían los siguientes:

- **CubafeaturesComp.Endpoints:** Misma función que el Endpoints del componente anterior, se asignarán los puertos en los que se enviarán las salidas del componente *CubafeaturesComp*.
- **LaserProxy:** Este parámetro servirá para fijar la dirección del puerto y el servidor o equipo remoto del cual se captarán los datos de salida del componente *laserRGBD* asociado. Es decir, se aplicará el algoritmo CUBA a un *scan* láser virtual concreto.
- **DiferentialRobotProxy:** Misma función exactamente que el DiferentialRobotProxy anterior, puesto que la información del robot utilizado se encontraría en el mismo puerto.
- **Height:** Este parámetro determinará la altura asignada al grupo de marcas extraídas de un *scan* láser concreto. Se asignará manualmente igual que los anteriores y se asignará típicamente el valor medio entre la MaxHeight y MinHeight fijadas en el componente láser asociado.

B.3. Parámetros Curvature3Dcomp

Igual que los anteriores parámetros el nombre del archivo de configuración asignado será el que queramos. La función de este algoritmo será agrupar todas las marcas extraídas de los diferentes *scan* láser, de manera que se crea una estructuras de datos con todas las marcas almacenadas y organizadas. Los parámetros serán los siguientes:

- **curvature3Dcomp.Endpoints:** Se asignará la dirección del puerto donde se enviarán los datos de salida del componente.
- **CubaNumber:** Se le asignará el valor del número de planos utilizados en la exploración. Viene también dada por el número de componentes *laserRGBD* y *CubafeaturesComp* ejecutados en paralelo y en tiempo real.
- **CubaProxyX:** Se asignará la dirección del puerto y el servidor o equipo remoto del cual se recibirán los datos de salida de cada uno de los componentes *cubafeaturesComp*. Es decir, para cada proxy se recibirán los datos de las marcas naturales extraídas en un *scan*.El valor de X variará para cada uno de los datos de salida provenientes de cada *scan*.
- **CubaIDX:** Se asignará el identificador de cada uno de los proxy asociados. La X variará con el número que se asignamos a cada proxy.

Bibliografía

[1] P. Núñez, R. Vázquez-Martín, J.C. del Toro, A. Bandera and F. Sandoval, "Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation". Robotics and Autonomous Systems, Vol 56, (3), pp. 24–264, 2008.

[2] A. Elfes,"Sonar-based real-world mapping and navigatio", IEEE Journal of Robotics and Automation 3(3), pp. 249-265,1987.

[3] B. J. Kuipers, "The spatial semantic hierarchy", Artificial Intelligence 119, pp. 191-233, 2000.

[4] A. Nuchter and J. Hertzberg, "Towards semantic maps for mobile robots", in Journal of Robotics and Autonomous Systems (JRAS), Special Issue on Semantic Knowledge in Robotics, vol. 56, no. 11, pp. 915-926, 2008.

[5] J. D. Tardós, J. Neira, P. M. Newman and J. J. Leonard, "Robust mapping and localization in indoor environments using sonar data", Int. Journal of Robotics Research, pp. 311-330, 2002.

[6] P. Henry, M. Krainin, E. Herbst, X. Ren and D. Fox. "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments" in International Symposium on Experimental Robotics, 2010.

[7] N. Muhammad, D. Fofi, and S. Ainouz, "Current state of the art of vision based SLAM" in Proceedings of the SPIE, article id. 72510F, 12 pp, 2009.

[8] V. Nguyen, S. Gachter, A. Martinelli, N. Tomatis, R. Siegwart, "A comparison of line extraction algorithms using 2D range data for indoor mobile robotics", in Autonomous Robots 23 (2) (2007) 97-111

[9] C. Fernández, V. Moreno, B. Curto, J.A. Vicente, "Clustering and line detection in laser range measurements", in Robotics and Autonomous Systems 58, pp. 720-726, 2010.

[10] M. Movafaghpour and E. Masehian, "Poly line map extraction in sensor-based mobile robot navigation using a consecutive clustering algorithm", in Robotics and Autonomous Systems, vol. 60, pp. 107–1092, 2012.

[11] Y. Zhao and X. Chen, "Prediction-based geometric feature extraction for 2D laser scanner", in Robotics and Autonomous Systems, Vol. 59, pp. 40–409, 2011.

[12] R. Madhavan, H.F. Durrant-Whyte, "Natural landmark-based autonomous vehicle navigation", Robotics and Autonomous Systems, Vol. 46, pp. 79-95, 2004.

[13] G.A. Borges, M. Aldon, Line extraction in 2D range images for mobile robotics, Journal of Intelligent and Robotic Systems Vol. 40, pp. 267-297. 2004.

[14] L.J. Manso, P. Bachiller, P. Bustos, P. Nuñez, R. Cintas and L. Calderita, "RoboComp: a Tool-based Robotics Framework". In Simulation, Modeling and Programming for Autonomous Robots (SIMPAR), pp. 251–262. 2010.

[15] G. Borges and M. Aldon. Line extraction in 2d range images for mobile robotics. Journal of Intelligent and Robotic Systems, 40:267?297, 2004.

[16] "A Multi-Layer Description of the Environment using Curvature Information for Robot Navigatio", J. Almeida, L. J. Manso, Antonio Bandera y Pedro Núñez. Presentado en el ?XV Workshop of Physical Agents? de León, España. (Junio 2014)

[17] http://robocomp.sourceforge.net/wordpress/

[18] M. Henning and M. Spruiell. "Distributed Programming with Ice". 2009.