INSTITUTO UNIVERSITARIO DE SISTEMAS INTELIGENTES Y APLICACIONES NUMÉRICAS EN INGENIERÍA
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

TRABAJO FINAL DE MÁSTER

ELABORACIÓN DE MAPAS COGNITIVOS DE ENTORNO EMPLEANDO UN SENSOR LÁSER ACTIVO

Víctor Prieto Marañón
Las Palmas de Gran Canaria - diciembre de 2009

Trabajo Final de Máster

Título: ELABORACIÓN DE MAPAS COGNITIVOS DE ENTORNO EMPLEANDO UN SENSOR LÁSER ACTIVO
Apellidos y nombre del alumno: Prieto Marañón, Víctor
Fecha : diciembre de 2009

Tutor: Cabrera Gómez, Jorge
Cotutor: Domínguez Brito, Antonio Carlos
Cotutor: Hernández Sosa, José Daniel

Por los ausentes

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Este trabajo tiene como objetivo experimentar con dos entornos tridimensionales de grandes dimensiones en los que se trata de construir un mapa de superficies multinivel (MSM) único y coherente a partir del cual se quiere detectar y etiquetar ciertas estructuras. Para ello, se usará un sistema robótico dotado de un sensor láser y un dispositivo apuntador. Los datos adquiridos del entorno tridimensional se integrarán en un único mapa tridimensional. Hemos usado para ello un mapa de superficies multinivel ya que, con un consumo de memoria limitado, permite una correcta identificación de las estructuras que, a diferentes alturas, existen en un espacio tridimensional

Se trabajará en grandes espacios en los que, bien por su dimensión o bien por existir oclusiones entre objetos, será necesario desplazar el sistema de adquisición de datos y tomar múltiples scans 3D. Estos múltiples scans deberán situarse en un único sistema de referencias, operación que denominaremos registro de los scans.

El registro de múltiples scans 3D requiere la localización y orientación exacta desde la que se hizo el scan. Sin embargo, la fuente habitual de localización y orientación en un sistema robótico, la odometría, carece de precisión. Esta imprecisión se va acumulando a medida que el robot se desplaza largas distancias, llegando a tener un gran impacto cuando se cierra un circuito, esto es, se vuelve a pasar por una zona ya visitada. La creación de un único mapa global requiere el uso de técnicas de SLAM (Simultaneous localization and mapping). Se han estudiado distintas soluciones, como la de M. Montemerlo, S. Trun, D. Killer y B. Wegbreit [1] y la de E. Olson, J. Leonard y S. Teller [2]. Basada en esta última, se ha desarrollado por parte de G. Grisetti, C. Stachniss, S. Grzonka y W. Burgard [3] una herramienta llamada «Toro» (Tree-based netwORk Optimizer)[1]. TORO es una solución para optimizar redes de restricciones basada en un procedimiento de minimización del error basada en descenso del gradiente. En nuestro trabajo hemos usado un sistema de SLAM con seis grados de libertad (6DoF) desarrollado por Andreas Nüchter et al [4] el cual nos permite trabajar en entornos tridimensionales.

Una vez creado un mapa en el que se han registrado todos los múltiples scans 3D, se pretende localizar, identificar y etiquetar en el mapa estructuras correspondientes a suelos, techos, paredes,

[1] http://www.openslam.org/toro.html

mesas y puertas. Todas las estructuras identificadas se recogen en un fichero de salida desde el que se pueden consultar. Además, se proporcionan unas herramientas de visualización en las que se puede observar el mapa de forma tridimensional y una vista 2D en la que se recogen las posibles puertas.

Un esquema del sistema desarrollado en este trabajo se puede ver en la figura 1.1. El sistema pasa por tres fases. En la primera (arriba a la izquierda en la figura) el sistema robótico es teledirigido a distintas localizaciones desde las que tomará scans tridimensionales del entorno. Los datos proporcionados por el sensor son convertidos en coordenadas espaciales y almacenadas en ficheros. En la segunda fase, se usa el software SLAM 6D para corregir las localizaciones y minimizar el error proporcionado por la odometría. Este software, almacena en ficheros la matriz de rotación y traslación de cada posición lo que permitirá registrar los distintos scans en un mapa único de forma coherente. Finalmente, la tercera fase construye un mapa de superficies multinivel (MSM) a partir de los datos proporcionados por los scans y de las matrices de transformación obtenidas a la salida del SLAM 6D. A partir del mapa, en esta fase se localizan y etiquetan ciertas estructuras del entorno. Como salida, se obtienen un fichero con la descripción de las estructuras encontradas, una ventana con una vista 3D del mapa y una ventana con una vista 2D cenital en la que se ubican las posibles puertas encontradas.

1.1. Estado actual

Comentaremos a continuación, brevemente, algunas tendencias en las técnicas de representación del entorno y de SLAM.

Una forma popular de afrontar el problema de crear mapas a partir de robots móviles en entornos abiertos han sido los mapas de elevación [5]. Estos mapas aplican una representación en $2\ \frac{1}{2}$ dimensiones. Un mapa de elevación consta de una malla de dos dimensiones en las que cada celda almacena la altura del terreno. Mientras que este acercamiento conlleva una reducción de memoria sustancial, puede ser problemática cuando un robot tenga que utilizar dichos mapas para navegar o cuando tenga que integrar dos mapas diferentes.

Una mejora que se ha propuesto a la solución anterior, y que se va a usar en este proyecto, es la propuesta por Triebel, Pfaff y Burgard en [6]. Básicamente, esta estrategia realiza una extensión de los mapas de elevación hacia múltiples superficies. Estos mapas multisuperficie ofrecen la posibilidad de modelar el entorno con más de un nivel transitable. La ventaja de esta solución es que se puede almacenar de forma compacta y, al mismo tiempo, usarse para caracterizar el entorno y dar soporte al problema de asociación durante el alineamiento de distintos mapas.

Conjuntamente a la elaboración del mapa, surge también el problema de ser capaces de extraer información de mayor nivel semántico de la representación del mundo elegida. Mientras que la visualización de nubes de puntos muy detalladas y complejas es actualmente posible, las capacidades

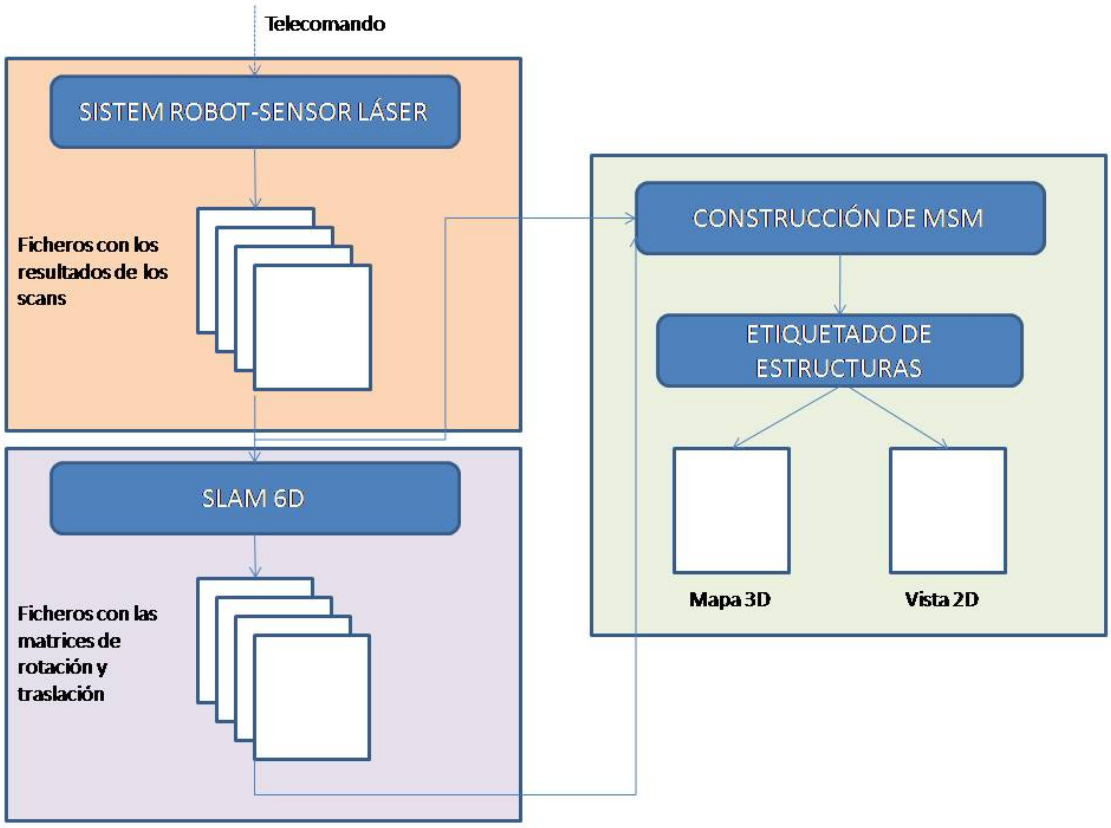

Figura 1.1: Esquema general del sistema desarrollado

de interacción en un cierto nivel semántico son aún muy limitadas. Incluso tareas tan básicas como seleccionar todas las ventanas en un scan de una casa requiere una cantidad desproporcionada de interacción con el usuario. Esto es debido al hecho de que los datos adquiridos durante el scan no proporcionan ninguna información estructural ni semántica, o en todo caso muy débil.

Sin embargo, el trabajar con nubes de puntos tiene una doble motivación. Por una parte, es la forma natural en la que se adquiere la información por parte de sensores láser o similares. Además, las nubes de puntos son representaciones geométricas extremadamente generales debido a que no contienen representaciones de los datos.

Se ha propuesto un gran número de métodos para la detección de formas primitivas. En la visión por ordenador, las dos metodologías mejor conocidas son el paradigma RANSAC [7] y la transformada de Hough [8]. Ambas han probado con éxito la detección de formas en 2D al igual que en 3D, siendo fiables incluso en presencia de un alto número de puntos ajenos, pero carecen de eficiencia debido al gran consumo de memoria. Distintos autores han desarrollado técnicas de aceleración. En este proyecto se implementa un versión del método Efficient RANSAC de Schnabel y Wahl [9] para detectar planos.

La localización y construcción simultánea de mapas (SLAM) es un problema que se define como un robot en un entorno desconocido y que parte de una localización de la que no conoce sus coordenadas. La trayectoria que sigue el robot es incierta, haciendo que cada vez sea más difícil determinar sus coordenadas globales. El problema de SLAM consiste en construir un mapa del entorno simultáneamente con la determinación de la posición relativa del robot en el mapa.

Una configuración habitual del SLAM implica asumir que el entorno posee distintos puntos característicos. Estos puntos, coomo jambas de puertas o esquinas de habitaciones, pueden ser proyectados en un mapa 2D caracterizados en forma de puntos del plano. A medida que el robot se mueve puede estimar su posición relativa mediante la localización y emparejamiento de los puntos característicos de mapas obtenidos en distintos instantes.

Los problemas de SLAM se diferencian en un cierto número de aspectos diferentes. Algunas distinciones habituales que se señalan en [10] son:

- Volumétrico frente a basado en características: En el SLAM volumétrico el mapa es muestreado con una resolución lo bastante alta para permitir una reconstrucción fotorealista del entorno. El SLAM basado en características extrae unas pocas características de los datos proporcionados por el sensor. En este caso, el mapa consiste únicamente en una colección de características. Las técnicas basadas en características son más eficientes pero sus resultados son inferiores a los de las volumétricas.
- Topológicos frente a métricos. Algunas técnicas de construcción de mapas recuperan únicamente una descripción cualitativa del entorno, la cual recoge las relaciones entre algunas localizaciones. A estos métodos se les conoce como topológicos. Un mapa topológico puede definirse sobre un conjunto de distintos sitios y un conjunto de relaciones cualitativas entre

ellos (por ejemplo, el sitio A es adyacente al B). Los métodos métricos de SLAM proporcionan información métrica acerca de la relación entre distintos lugares (por ejemplo, A está a 5 m. de B). En los últimos tiempos, los métodos topológicos han caido en desuso.

- Correspondencia conocida frente a correspondencia desconocida. El problema de correspondencia es el problema de identificar como una misma entidad percepciones tenidas en distintos momentos. Algunos algoritmos de SLAM asumen que se conoce la identidad de los puntos característicos del entorno, otros no.

- Estático frente a dinámico. Los algoritmos estáticos asumen que el entorno no cambia a lo largo del tiempo. Los métodos dinámicos, por el contrario, parten de que el mundo cambiará.

- Pequeña frente a gran incertidumbre. Los problemas de SLAM se distinguen por el grado de incertidumbre en la localización que manejan. Los algoritmos más simples sólo permiten errores pequeños en la estimación de la localización. Son adecuados en situaciones en las que un robot sigue una trayectoria que no se corta a si misma y al final vuelven por el mismo camino. En ciertas ocasiones, el robot puede retornar al punto de origen siguiendo caminos muy diversos. En dichos casos, el robot puede acumular una gran cantidad de incertidumbre a lo largo de su trayectoria. Esto se conoce como el problema de bucle cerrado.

- Activo frente a pasivo. En los algoritmos de SLAM pasivos, el control del robot es llevado a cabo por una entidad ajena, y el algoritmo de SLAM se limita a observar. La gran mayoría de los algoritmos son de este tipo. En los algoritmos activos, el robot es el que explora el entorno con el objetivo de construir el mapa.

- Robot simple frente a robot múltiple. La mayoría de los problemas de SLAM están diseñados para un único robot. Recientemente, sin embargo, el problema de exploración con múltiples robots ha ganado popularidad.

1.2. Estructura de la memoria

La memoria se divide en los siguientes capítulos:

1.- Introducción. El presente capítulo, desde el que se resume el trabajo desarrollado.

2.- Adquisición de datos. En dicho capítulo se expone el hardware usado, se describen los entornos elegidos como escenarios de pruebas y se explica cómo se generan las coordenadas espaciales de los objetos del entorno a partir de la información proporcionada por el sistema robótico-sensor láser.

3.- SLAM 6D. El tercer capítulo está dedicado a presentar el algoritmo de Nüchter et al. para SLAM tridimensional con seis grados de libertad. En la parte final del capítulo se comenta los parámetros del programa SLAM 6D de los mismos autores.

4.- Etiquetado semántico de elementos del entorno. El capítulo 4 explica cómo se identifican y localizan suelos, puertas, paredes, techos y tableros.

5.- Resultados. En este capítulo se exponen los resultados obtenidos en los dos escenarios de pruebas elegidos.

6.- Conclusiones y trabajo futuro. El último capítulo expone las conclusiones del trabajo y señala posibles líneas de trabajo futuro.

Capítulo 2

Adquisición de datos

En este trabajo se han usado dos escenarios distintos. Uno ha sido el pasillo del sótano del Edificio Central del Parque Tecnológico de la ULPGC, el otro escenario ha sido el laboratorio de Robótica situado en el mismo edificio. Ambos entornos tienen características diferentes. El primero es de grandes dimensiones y con pocas zonas de oclusión más allá de los propios recodos del pasillo. El laboratorio de Robótica, por su parte, es de dimensiones más reducidas y cuenta con numerosas zonas de oclusión debido a los numerosos objetos que allí se encuentran: armarios, mesas, sillas, ordenadores, paneles, etc. En el resto del capítulo describiremos el sistema de adquisición de scans, comentaremos con detalle cómo ha sido la recogida de datos en los dos escenarios y, finalmente, se describirá cómo se genera la nube de puntos a partir de las lecturas del sensor láser. El sistema de generación de coordenadas espaciales está basdo en el proyecto final de carrera [11].

2.1. Sistema de adquisición de scans

El sistema de adquisición de scans ha estado formado por un robot Pioneer P3-DX, un sensor láser Hokuyo UTM-30LX, un dispositivo apuntador Directed Perception PTU 46-17.5, un joystick inalámbrico de Logitech y un ordenador personal como unidad de procesamiento y control.

El sensor láser va montado sobre el dispositivo apuntador y ambos, a su vez, sobre el robot (ver fig. 2.1). El sistema es teledirigido a través del joystick inalámbrico. Mediante los botones y palancas del joystick se puede desplazar el robot y se puede ordenar la adquisición de un scan 3D.

El robot Pioneer P3-DX sirve como base al sistema sensor láser-dispositivo apuntador y como dispositivo de desplazamiento. En este caso, el movimiento del robot será teledirigido para situarlo en las posiciones desde las que interese tomar scans 3D. Este robot posee unas dimensiones de 44.5cm de largo, 40cm de ancho y 24.5cm de alto. El robot es capaz de suministrar una odometría que será usada como estimación inicial de la posición del robot.

El sensor láser, de la empresa Hokuyo, posee un gran rango de medición lo que lo hace apto para trabajar en entornos de grandes dimensiones. Su pequeño peso y dimensiones lo hacen adecuado

Figura 2.1: Sistema formado por el Robot Pioneer P3-DX, el sensor láser Hokuyo UTM-30LX y el dispositivo apuntador Directed Perception PTU 46-17.5. El sensor láser y el dispositivo apuntador se observan sobre la plataforma a la izquierda. A la derecha de ambos, el joystick inalámbrico

Tabla 2.1: Especificaciones del Sensor Láser Hokuyo UTM-30LX

Fuente de energía	12V DC ± 10 %Consumo máx: 1A, normal: 0.7A
Fuente de luz	Diodo láser semiconductor ($\lambda = 785nm$). Seguridad láser clase 1
Rango de detección	$0{,}1 \approx 30m$ Máx 60m. 270°
Precisión	$0{,}1 \approx 10m : \pm 30mm$
	$10 \approx 30m : \pm 50mm$
Resolución angular	$0{,}25°(360°/1440steps)$
Tiempo de scan	$25ms/scan$ (40 Hz.)
Interfaz	USB 2.0
Peso	370 g.

Tabla 2.2: Unidad PTU-D46-17.5

Peso	1.38 kg
Dimensiones	7,62cm ancho
	13.03cm alto
	10.795cm fondo
Velocidad máxima sin carga	300^o por segundo
Resolución	0.013^o
Rango de elevación	$-90^o a 31^o (78^o)$
Rango de azimut	$\pm 159^o$

para ser montado con facilidad sobre otros dispositivos, como en este caso. Vemos en la tabla 2.1 las principales características del sensor.

El dispositivo apuntador va a servir para dotar al sistema de la capacidad de obtener información del entorno tridimensional. Se ha usado en este trabajo un dispositivo apuntador de la empresa de Directed Perception, modelo PTU 46-17.5

Este dispositivo apuntador posee dos grados de libertad: uno horizontal (pan) y otro vertical o de azimut (tilt). Sobre la estructura que se ve en la parte superior se montará el sensor. Vemos algunas características del modelo en la tabla 2.2.

La característica esencial de este dispositivo apuntador es que proporciona un control preciso de la velocidad y aceleración de los ejes en base a dos motores paso a paso. Como se puede ver en la figura 2.2, los extremos entre los que puede oscilar una velocidad no estacionaria están determinados por la velocidad límite superior y la velocidad límite inferior. La velocidad base, o velocidad inicial, se refiere a la velocidad a la cual los ejes pueden ser iniciados partiendo de una parada absoluta sin pérdida de sincronía. Debido a los requisitos de la velocidad base y a que el par motor disminuye a medida que se incrementa la velocidad, se necesita aceleración para alcanzar velocidades por encima del nivel base. El controlador de la unidad usa aceleración y frenado trapezoidal para velocidades comprendidas entre la base y el máximo permitido.

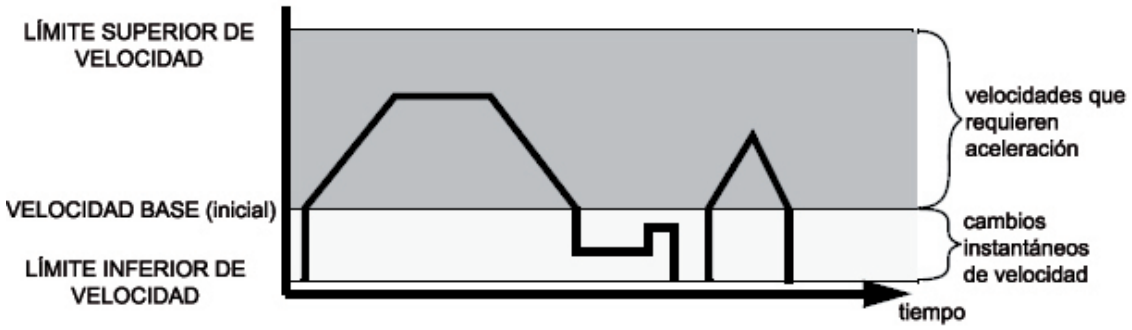

Figura 2.2: Velocidad en los ejes, Velocidades instantáneas, Aceleración trapezoidal y Cambios Instantáneos de velocidad y posición

La figura 2.2 muestra dos casos en los que se produce aceleración. En el primero, uno de los ejes acelera hasta alcanzar una velocidad constante deseada y, posteriormente, frena. En el segundo caso, vemos la situación en que la unidad no tiene tiempo suficiente para acelerar hasta una velocidad constante antes de que necesite frenar para llegar a la posición deseada. Este esquema se usará para estimar la posición del dispositivo apuntador en un determinado instante de tiempo. De esta forma, podremos localizar con precisión la situación de un objeto señalado por el sensor láser.

2.1.1. Sincronización entre el sensor láser y el dispositivo apuntador

Se desea mover el dispositivo apuntador a la máxima velocidad que permita obtener un scan tridimensional con una resolución determinada. Si no se quiere perder precisión en la localización espacial de los objetos del entorno, se debe elaborar un algoritmo que permita conocer la orientación del sensor láser en cada instante mientras se mueve de forma continua el dispositivo apuntador. Se usa en este trabajo el algoritmo desarrollado en [11]. Este algoritmo toma en cuenta la posición del dispositivo apuntador al inicio de cada barrido del láser y estima el ángulo de elevación vertical para cada medida proporcionada por el sensor. Podemos ver el algoritmo en la figura 2.3. En este algoritmo, en $t\_inicial$ se almacena el instante en que empieza a desplazarse el dispositivo apuntador. Este instante se obtiene promediando los instantes anterior y posterior a la llamada al comando que inicia el movimiento del dispositivo. Este promediado corrige las latencias que se producen entre el instante en que se invoca al comando y el instante en que éste rotorna. Posteriormente, se procede a ir adquiriendo mediciones del sensor láser. El sensor incluye el instante (*timestamp*) en que comenzó el barrido en el array que devuelve tras cada rotación del haz. Como en un barrido el haz láser pasa por 1440 posiciones, para una frecuencia f dada, el instante de tiempo en que se tomó la lectura l_p, siendo p el paso de la lectura, se calcula como:

$$t(p) = timestamp + \frac{p}{1440 \cdot f}$$

Por tanto, el tiempo transcurrido desde el inicio del desplazamiento del sensor hasta que se tomó la lectura l_p ha sido $t(p) - t\_inicial$. Con este lapso de tiempo se puede calcular la posición

del dispositivo apuntador tomando en cuenta el esquema de aceleración trapezoidal que se muestra en 2.2. El algoritmo que calcula la posición se muestra en la figura 2.4. El procedimiento discrimina tres posibles casos:

1. La velocidad de desplazamiento es inferior o igual a la velocidad base. En este caso el dispositivo apuntador no requiere aceleración, por lo que la velocidad es constante y la posición se estima mediante una interpolación lineal.

2. La velocidad de desplazamiento es superior a la velocidad base y, antes de terminar de acelerar, se alcanza la posición media de desplazamiento. En este caso, el dispositivo apuntador sigue un esquema de aceleración triangular. Se calcula si el instante de tiempo para el que se va a calcular la posición es anterior o posterior al vértice del triángulo y se obtiene la posición buscada.

3. La velocidad de desplazamiento es superior a la velocidad base y se termina de acelerar antes de alcanzar la posición media de desplazamiento. Aquí el esquema de aceleración es trapezoidal. Habrá que calcular si el instante de tiempo para el que se quiere obtener la posición está situado sobre la rampa ascendente de aceleración, sobre la cima horizontal de velocidad constante o sobre la rampa descendente de frenado. En función de la situación, se calcula la posición del dispositivo apuntador.

2.2. Escenarios de pruebas

Los scans 3D tomados en los dos escenarios de prueba fueron realizados con una resolución azimutal de 0,5° y el barrido se efectuó entre los $-20°$ y los 20°. Cada scan 3D se almacenó en un fichero para su posterior procesamiento. En este fichero, además de los datos proporcionados por el sensor láser se almacenaban los datos necesarios para la posterior construcción del mapa 3D.

2.2.1. Pasillo

El pasillo es un escenario amplio con muy pocas zonas ocultas que no sean los propios recodos. Sólo producen oclusiones unas macetas dispuestas a lo largo del mismo. Las dimensiones aproximadas del pasillo son 40,5m. de largo por 4,75m. de ancho. El pasillo posee algunos recodos perpendiculares al mismo. La longitud del rocodo más largo llega a los 11m. En este escenario se tomaron 24 scans. En la imagen 2.5 tenemos una foto del pasillo y en la imagen 2.6 se muestra una planta del mismo con escala.

2.2.2. Laboratorio

El laboratorio, a diferencia del pasillo, es un escenario bastante más pequeño y con multitud de objetos que producen oclusiones. Las dimensiones aproximadas del laboratorio son de 8,3m. de ancho por 11,4m. de largo. En este caso se tomaron 8 scans 3D. En la imagen 2.7 podemos ver una foto del laboratorio y en la imagen 2.8 se muestra una planta del mismo con escala.

2.3. Obtención de coordenadas 3D

Una vez conocida la posición del dispositivo apuntador para cada una de las medidas suministradas por el láser, queda el problema de convertir estas medidas en coordenadas espaciales del mundo. Esta tarea será resuelta mediante las herramientas de la cinemática robótica. Antes de afrontar el problema cinemático directo debemos modelar nuestro sistema como una cadena cinemática. Una cadena cinemática, como sabemos, es un conjunto de eslabones y articulaciones interconectados de modo que proporcionan un movimiento de salida controlado en respuesta a una orden de entrada. Los eslabones son una serie de elementos estructurales sólidos. A su vez, cada articulación proporciona, al menos, un grado de libertad. Existen dos tipos de articulaciones: prismáticas y rotacionales. La articulación prismática es aquella en la que el eslabón se apoya en un deslizador lineal. La articulación rotacional es tal que el eslabón realiza un giro pivotando sobre la propia articulación. En nuestro sistema sólo dispondremos de articulaciones rotacionales. En concreto, nuestro sistema dispone de tres articulaciones. Las dos primeras articulaciones las proporciona el dispositivo apuntador con sus dos motores. La tercera articulación será el motor del sensor considerando al propio haz láser como un eslabón más. Efectivamente, nuestro interés radica en localizar espacialmente las coordenadas del punto señalado por el sensor. Por ello, aunque el haz láser no sea un eslabón al uso, sólido, lo consideramos como un eslabón más de la cadena el cual, en cada punto detectado, vendrá determinado por el ángulo del haz y la distancia del punto alcanzado.

Denavit y Hartenberg [12] propusieron en 1955 un método sistemático para describir y presentar la geometría espacial de los elementos de una cadena cinemática, y en particular de un robot, con respecto a un sistema de referencia fijo. Este método utiliza una matriz de transformación homogénea para describir la relación espacial entre dos elementos rígidos adyacentes, reduciéndose el problema cinemático directo a encontrar una matriz de transformación homogénea cuatro por cuatro que relacione la localización espacial del extremo del robot con respecto al sistema de coordenadas de su base. Según la representación de Denavit-Hartenberg, escogiendo adecuadamente los sistemas de coordenadas asociados para cada eslabón, será posible pasar de uno al siguiente mediante cuatro transformaciones básicas que dependen exclusivamente de las características geométricas del eslabón.

Presentamos a continuación un algoritmo basado en el método de Denavit y Hartenberg para resolver el problema cinemático directo. El algoritmo se estructura en ocho pasos consecutivos:

1. Se localiza cada uno de los ejes de las articulaciones y se etiquetan como $z_0, \ldots, z_{n-1}$
2. Se establece el sistema de coordenadas base. Se sitúa el origen en cualquier punto del eje z_0. Los ejes x_0 e y_0 son elegidos convenientemente para que el sistema sea dextrógiro. Los pasos 3, 4 y 5 se realizan para $i = 1, ..., n-1$
3. Se ubica el origen de coordenadas O_i en la intersección entre el eje z_i y la normal común a z_i y z_{i-1}. Si z_i y z_{i-1} son paralelos, se ubica O_i en cualquier punto que convenga de z_i
4. Se establece x_i a lo largo de la normal común entre z_{i-1} y z_i a través de O_i, o en la dirección normal al plano z_{i-1}, z_i si z_{i-1} y z_i se intersectan

5. Se establece y_i para que el sistema resultante sea dextrógiro.
6. Se localiza el n-ésimo sistema de coordenadas en el extremo del robot. En nuestro sistema lo localizaremos en el extremo del haz láser.
7. Se crea una tabla de parámetros de cada eslabón $a_i, d_i, \alpha_i, \theta_i$.
   - a_i = distancia a lo largo de x_i desde O_i hasta la intersección de los ejes x_i y z_{i-1}.
   - d_i = distancia a lo largo de z_{i-1} desde O_{i-1} hasta la intersección de los ejes x_i y z_{i-1}.
   - α = ángulo entre z_{i-1} y z_i medido en x_i.
   - θ = ángulo entre x_{i-1} y x_i medido en z_{i-1}.
8. Se forman las matrices de transformación A_i sustituyendo los cuatro parámetros anteriores.

Estas matrices de transformación consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento i con el sistema de referencia del elementos i-1. Las transformaciones en cuestión se componen de dos rotaciones y dos traslaciones en el siguiente orden:

1. Rotación alrededor del eje Z_{i-1} un ángulo θ_i;
2. Traslación a lo largo de Z_{i-1} una distancia d_i;
3. Traslación a lo largo de X_i una distancia a_i;
4. Rotación alrededor del eje X_i un ángulo α_i;

De este modo, la transformación del sistema de referencia i-1 al sistema i viene dado por la composición de las cuatro transformaciones anteriores. Expresadas cada una de las cuatro mediante matrices homogéneas, se tiene:

$$R_{z,\theta_i} = \begin{bmatrix} cos(\theta_i) & -sen(\theta_i) & 0 & 0 \\ sen(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trasl_{z,d_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trasl_{x,a_i} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tabla 2.3: Parámetros de la cadena cinética dispositivo apuntador - sensor láser

Eslabón	θ_i	d_i	a_i	α_i
Eslabón 1	Ángulo horizontal	h_0 mm	0mm	$-\pi/2$
Eslabón 2	-Ángulo vertical	0mm	-50mm	$\pi/2$
Eslabón 3	Ángulo del haz láser	100mm	Rango del haz (mm)	0^o

$$R_{x,\alpha_i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sen(\alpha_i) & 0 \\ 0 & sen(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

El producto de las cuatro transformaciones nos proporciona la matriz A_i de transformación del sistema i-1 al sistema i.

$$A_i = \begin{bmatrix} cos(\theta_i) & -sen(\theta_i)cos(\alpha_i) & sen(\theta_i)sen(\alpha_i) & a_icos(\theta_i) \\ sen(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sen(\alpha_i) & a_isen(\theta_i) \\ 0 & sen(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Finalmente, el producto de las distintas matrices de transformación correspondientes a cada eslabón nos permitirán expresar la localización espacial del extremo de la cadena cinemática en coordenadas referidas al sistema de referencia base.

Como ya comentamos, nuestra cadena cinemática está formada por el dispositivo apuntador y el haz láser. Dispondremos por tanto de tres eslabones, dos «físicos», proporcionados por la estructura del apuntador, y un tercer eslabón «lumínico», el propio haz láser. Las dos primeras articulaciones las forman los motores horizontal y vertical del apuntador y la tercera articulación, el motor del sensor láser (ver fig. 2.9). Para poder caracterizar nuestro sistema, debemos indicar los valores de los cuatro parámetros a_i, d_i, α_i y θ_i. El origen del sistema de coordenadas base lo situaremos, por conveniencia, en el suelo. La distancia h_0 desde este origen de coordenadas O_0 hasta O_1 se ajustará según dónde se monte el dispositivo apuntador sobre el robot. De esta forma, la distancia entre ambos sistemas de referencia será la altura del primer eslabón del dispositivo apuntador sobre el suelo. El origen del sistema de coordenadas O_1 está situado en el eje de la articulación vertical. El siguiente origen de coordenadas, O_2 se sitúa debajo del sensor, en la intersección del eje del motor del sensor con la normal al eje vertical. Finalmente, el origen último sistema de coordenadas O_3 se halla en el extremo del haz láser, hasta dondequiera que éste se extienda. Para poder ajustar los parámetros a_i y d_i se recurre a los datos de tamaño proporcionados por los fabricantes.

De esta forma, sea H el ángulo formado por el motor horizontal, V el ángulo formado por el motor vertical, L el ángulo del haz láser, R la longitud del haz y h_0 la altura del primer eslabón

sobre el suelo, la matrices de transformación A_i correspondientes a cada uno de los eslabones de nuestro sistema quedan de la siguiente forma:

$$A_1 = \begin{bmatrix} cos(H) & 0 & -sen(H) & 0 \\ sen(H) & 0 & cos(H) & 0 \\ 0 & -1 & 0 & h_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} cos(V) & 0 & -sen(V) & -50 \cdot cos(V) \\ -sen(V) & 0 & -cos(V) & 50 \cdot sen(V) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} cos(L) & -sen(L) & 0 & R \cdot cos(L) \\ sen(L) & cos(L) & 0 & R \cdot sen(L) \\ 0 & 0 & 1 & 100 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
mover apuntador(posición inicial)
esperar finalización movimiento apuntador
modificar velocidad apuntador(resolución deseada)
t1 := obtener hora
mover apuntador(posición final)
t2 := obtener hora
t_inicial := (t1+t2)/2
repetir
    medidas := obtener siguiente scan del láser
    /*
       Almacenamos en t_scan el instante en que se inició el último barrido del
       sensor. El sensor devuelve este dato en el array de medidas
    */
    t_scan := medidas.timestamp
    para cada lectura en medidas hacer
       tilt_angle := aceleración_trapezoidal(velocidad, velocidad base,
       aceleración, t_scan-t_inicial, ángulo inicial, ángulo final)
       t_scan := t_scan + 1/(frecuencia * 1440)
       procesar(lectura, tilt_angle)
    fin para
    tilt := obtener del dispositivo apuntador el ángulo tilt
hasta que tilt < posición final
```

Figura 2.3: Algoritmo de adquisición de datos que adquiere datos del láser mientras se realiza un movimiento contínuo del dispositivo apuntador con estimación precisa de la posición en cada instante. El algoritmo hace uso del esquema de aceleración trapezoidal del dispositivo apuntador

```
entrada:
vd - velocidad programada
vb - velocidad base
a - aceleración
t - instante para el que se desea calcular el ángulo de elevación
tilt0 - ángulo de elevación inicial
tiltf - ángulo de elevación final
salida:
tilt - ángulo de elevación en el instante t
si vd<vb entonces
     tilt=tilt0+vd*t
     retornar
si no
     ta=(vd-vb)/a
     tilta=tilt0+vb*ta+a*(ta^2)/2
     si tilta>(tilt0+tiltf)/2 entonces
          tm=(-vb+raiz(vb^2-2*a*(tilt0-tiltf)/2))/a
          si t<tm entonces
               tilt=tilt0+vb*t+a*t^2/2
               retornar
          si no
               tiltm=tilt0+vb*tm+a*tm^2/2
               vm=vb+a*tm
               tilt=tiltm+vm*(t-tm)-a*(t-tm)^2/2
               retornar
          fin si
     si no
          si t<ta entonces
               tilt=tilt0+vb*t*a*t^2/2
               retornar
          si no
               tb=ta+(tiltf-2*tilta)/vd
               si t<tb entonces
                    tilt=tilta+vd*(t-ta)
                    retornar
               si no
                    tiltb=tilta+vd*(tb-ta)
                    tilt=tiltb+vd*(t-tb)-a*(t-tb)^2/2
                    retornar
               fin si
          fin si
     fin si
fin si
```

Figura 2.4: Algoritmo que proporciona el ángulo de inclinación vertical en función del tiempo transcurrido desde el comienzo del desplazamiento del dispositivo apuntador.

Figura 2.5: Escenario de pruebas 1: Pasillo del sótano del Edificio Central del Parque Tecnológico. ULPGC

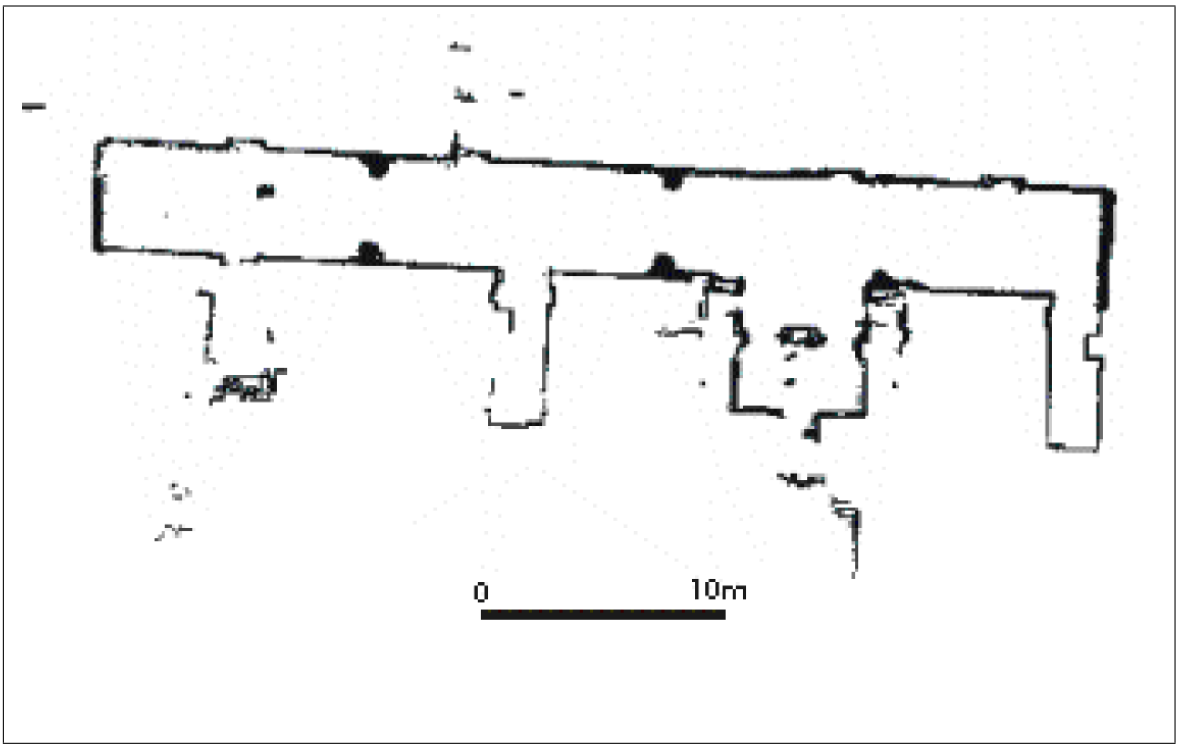

Figura 2.6: Escenario de pruebas 1: Planta del pasillo del sótano del Edificio del Parque Tecnológico con escala

Figura 2.7: Escenario de pruebas 2: Laboratorio de robótica del Edificio Central del Parque Tecnológico. ULPGC

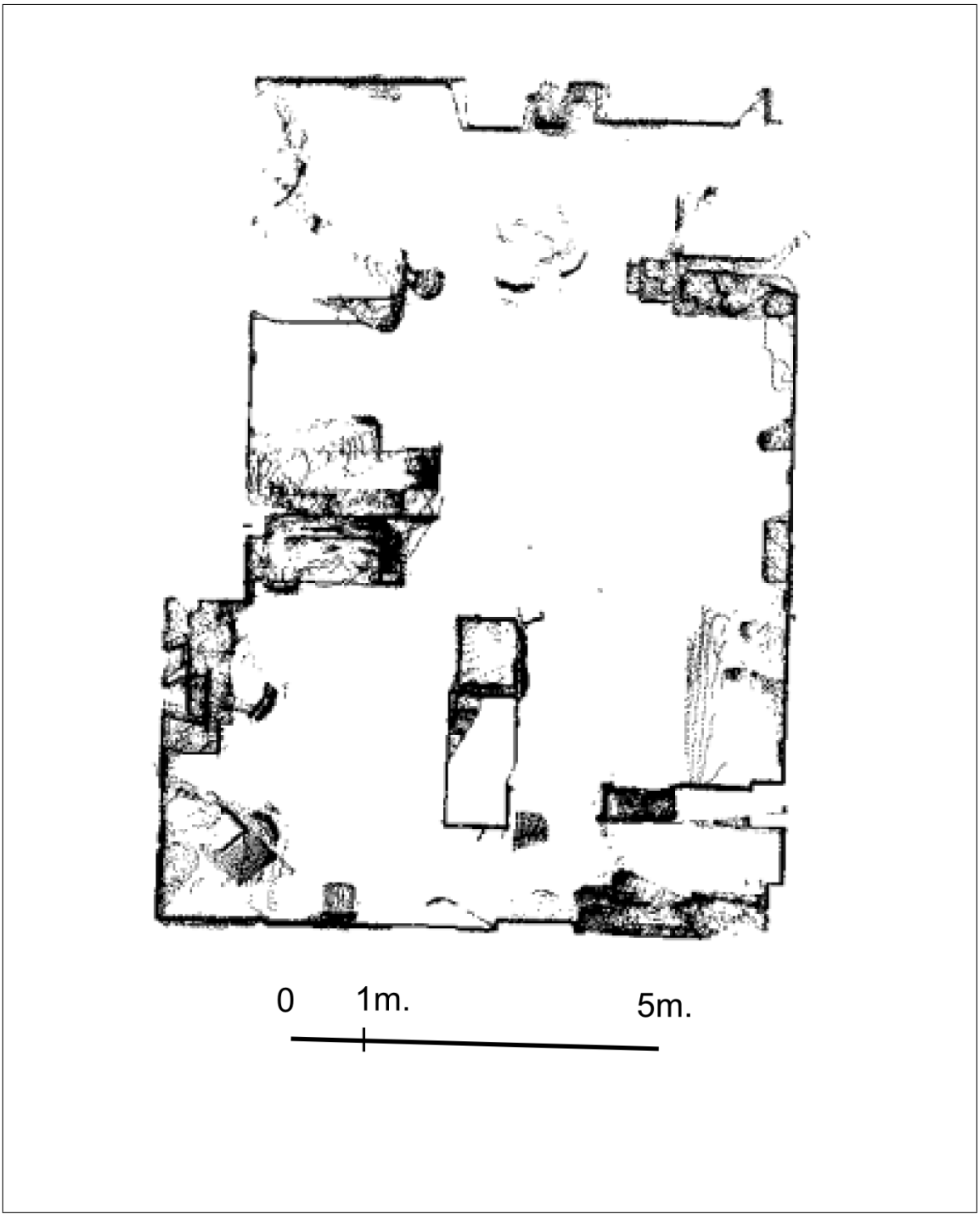

Figura 2.8: Escenario de pruebas 2: Planta del laboratorio de Robótica con escala

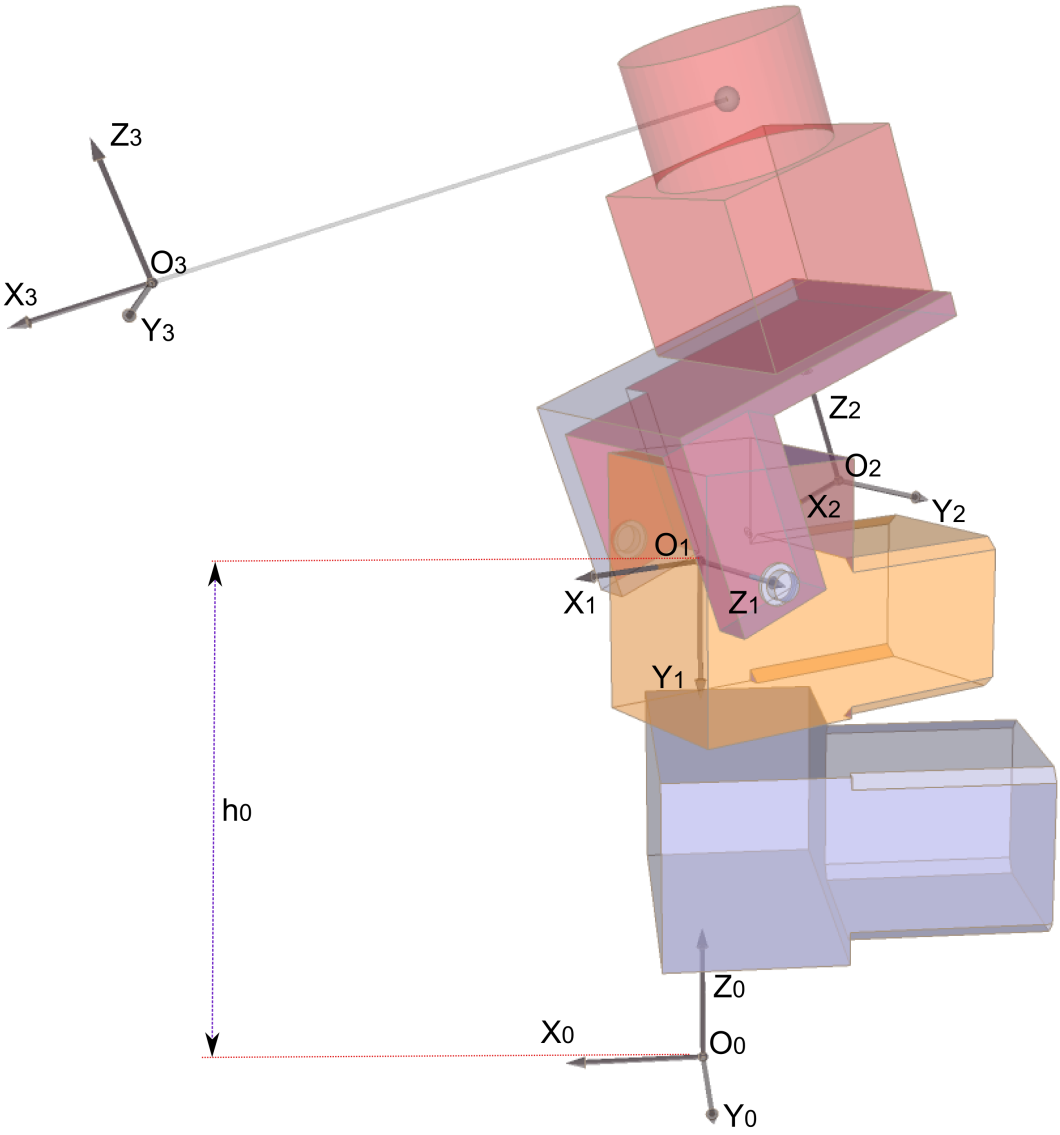

Figura 2.9: Representación de los ejes de referencia del sistema como cadena cinemática

Capítulo 3

SLAM 6D

La digitalización de entornos en los que existen zonas ocultas por algún o algunos objetos (paredes, armarios, mesas, etc.) hace necesario que se tengan que tomar varios scans 3D. Estos scans deberán ser combinados en un único sistema de coordenadas con el fin de obtener un único mapa consistente. A este proceso, la combinación de múltiples scans en un único mapa, se le llama registro. Si la localización del robot que transporta el escáner 3D fuera exacta, el registro podría hacerse directamente a partir de la posición del robot. Sin embargo, los sensores de posición de los robots tienen un nivel significativo de imprecisión, con lo que la autolocalización es errónea. Por tanto, cuando se procede al registro de los múltiples scans 3D hay que tener en cuenta la estructura geométrica de las zonas donde se solapan los distintos scans 3D.

Una representación globalmente consistente del entorno de un robot es crucial para distintas tareas como la localización y la navegación. Existen múltiples sistemas móviles que, equipados con un escáner láser, recogen información de su entorno local. Dichas representaciones locales tiene que ser emparejadas entre sí para poder crear un mapa global. Las aplicaciones iterativas que trabajan emparejando pares de scans llevan a inconsistencias debido a errores en la medida y al procedimiento de emparejamiento en si mismo. Para evitar dichos problemas, se necesita de un algoritmo de emparejamiento global, que tome en cuenta las correspondencias entre todos los scans. Los métodos más comunes para combinar todos los scans están basados en información probabilística. Lu y Milios presentaron un solución [13] que hace uso de una red de relaciones entre las posiciones del scan láser. Un sistema de ecuaciones lineales, construido a partir de todos los errores medidos, produce una estimación óptima para cada posición. Esta solución, restringida a un scan láser 2D, no satisfará los requisitos de construir un mapa correcto en entornos de exterior. Borrmann, Elseberg, Lingemann, Nüchter y Hertzberg han presentado una extensión del algoritmo de estimación lineal para que trabaje con scans 3D y con seis grados de libertad [14]. Los autores integran el método de optimización global en un sistema que consta de extrapolación odométrica y del conocido algoritmo iterativo del punto más cercano (ICP). Podemos ver el esquema general del algoritmo en la figura 3.1

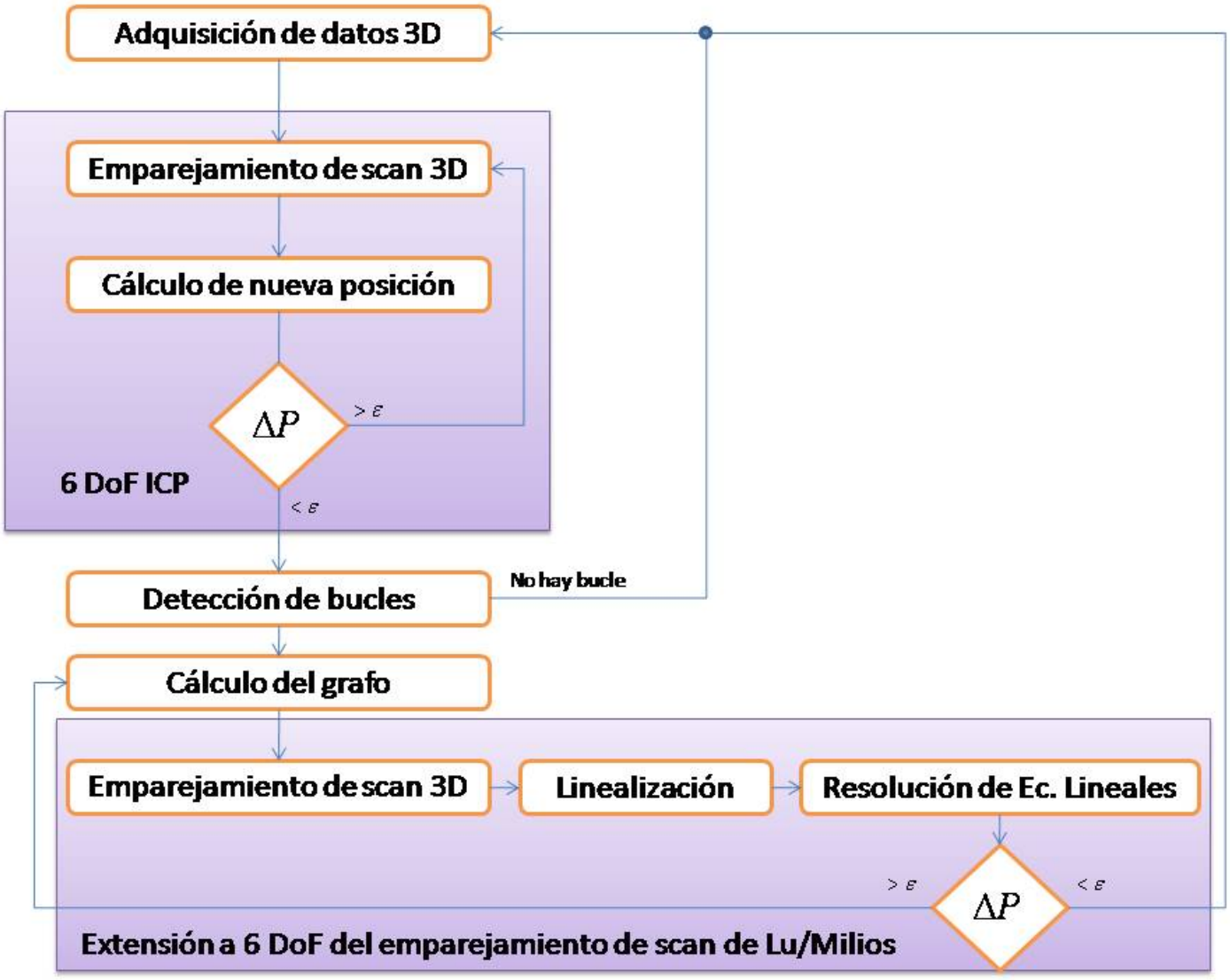

Figura 3.1: Algoritmo completo de SLAM 6D

3.1. Extensión a seis grados de libertad

Debido a que casi todos los robots móviles tienen un odómetro que les permite medir la distancia desplazada, el algoritmo desarrollado por Nüchter et al. usa dichas medidas odométricas para calcular una primera estimación de la posición [4]. La odometría debe ser extrapolada a seis grados de libertad y esto se hace usando las matrices de registro previas. De este modo, el cambio en la posición del robot ΔP, dada la información odométrica $(x_n^{odo}, y_n^{odo}, \Theta_{z,n}^{odo}), (x_{n+1}^{odo}, y_{n+1}^{odo}, \Theta_{z,n+1}^{odo})$ y la matriz de registro $R(\Theta_{x,n}, \Theta_{y,n}, \Theta_{z,n})$[1], se calcula resolviendo:

$$\begin{pmatrix} x_{n+1}^{odo} \\ y_{n+1}^{odo} \\ \Theta_{z,n+1}^{odo} \end{pmatrix} = \begin{pmatrix} x_n^{odo} \\ y_n^{odo} \\ \Theta_{z,n}^{odo} \end{pmatrix} + \left(\begin{array}{c|ccc} R(\Theta_{x,n}, \Theta_{y,n}, \Theta_{z,n}) & 0 & & \\ \hline & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ & 0 & 0 & 1 \end{array}\right) \cdot \underbrace{\begin{pmatrix} \Delta x_{n+1} \\ \Delta y_{n+1} \\ \Delta z_{n+1} \\ \Delta\Theta_{x,n+1} \\ \Delta\Theta_{y,n+1} \\ \Delta\Theta_{z,n+1} \end{pmatrix}}_{\Delta P} \tag{3.1}$$

Finalmente, la posición 6D P_{n+1} es calculada, usando la representación de matriz de posición, como:

$$P_{n+1} = \Delta P \cdot P_n \tag{3.2}$$

De esta forma, la odometría plana es extrapolada a seis grados de libertad al plano (x, y) definido por la última posición del robot.

3.2. El algoritmo ICP

Comentaremos brevemente, por ser actualmente ampliamente conocido, el algoritmo ICP. El algoritmo ICP (Iterative Closest Points) es usado para registrar nubes de puntos. El algoritmo fue inventado en 1991 simultáneamente por Besl y McKay, Chen y Medioni y por Zhang [15].

A partir de dos conjuntos de puntos, el conjunto modelo $\widehat{M}(\left|\widehat{M}\right| = N_m)$ y el conjunto de datos $\widehat{D}(\left|\widehat{D}\right| = N_d)$, el algoritmo trata de encontrar la transformación (R, t), formada por una matriz de rotación R y por un vector de traslación t, que minimice la siguiente función de coste:

$$E(R,t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} \omega_{i,j} \left\| \widehat{m}_i - (R\widehat{d}_j + t) \right\|^2 \tag{3.3}$$

En esta función, $\omega_{i,j}$ vale 1 si el i-ésimo punto de $\widehat{M}$ describe el mismo punto del espacio que el j-ésimo punto de $\widehat{D}$. En caso contrario, $\omega_{i,j}$ vale cero.

[1] El algoritmo de Nüchter et al. usa un sistema de coordenadas levógiro. Nosotros, en el presente trabajo, hemos usado un sistema dextrógiro

```
Para i=0 hasta maxIterations hacer
    Para todo d_j ∈ D hacer
      Encontrar el punto más cercano a d_j, dentro de un rango d_max en M
    Fin para
    Calcula la transformación (R, t) que minimiza la función de error
    Aplicar la transformación del paso anterior al conjunto D
    Calcular la diferencia del error cuadrático antes y después de la
    aplicación de la transformación.
    Si esta diferencia es menor que un umbral ε, termina.
Fin para
```

Figura 3.2: Algoritmo ICP

Se debe calcular, por una parte, la correspondencia entre los puntos de ambos conjuntos y, por otra, la transformación (R, t) que minimice $E(R, t)$.

El algoritmo ICP calcula iterativamente la correspondencia entre los puntos (ver fig. 3.2. En cada iteración, el algoritmo selecciona los puntos más cercanos como los correspondientes entre ambos conjuntos. Asimismo, en cada iteración calcula la transformación (R, t) que minimiza la ecuación 3.3. Se asume que en la última iteración la correspondencia entre los puntos es correcta.

Besl y al. han demostrado que el método alcanza un mínimo. Sin embargo, el teorema no se cumple en este caso debido a la restricción introducida al establecer una distancia tolerable máxima, d_{max}, para asociar puntos. Este umbral se hace necesario debido a que los scans 3D se solapan sólo parcialmente. Después de una transformación, algunos puntos que no tenían un punto al que asociarse dentro del límite puede que si lo tengan y, por tanto, el valor de la función de error puede incrementarse. Finalmente, el algoritmo minimiza la ecuación 3.3.

Existen distintos métodos para minimizar la función de error [4]. Las distintas estrategias se dividen en métodos indirectos y directos. Dentro de los indirectos encontramos la simulación de un sistema físico de resortes. En cuanto a los métodos directos, también llamados cerrados, se conocen cuatro: usando descomposición de valores singulares, usando matrices ortonormales, usando cuaterniones unitarios y usando cuaterniones duales. También existe un método para encontrar una solución aproximada mediante un movimiento helicoidal debida a Hofer y Pottmann y aplicada al registro de scans por Rusu et al.

3.3. Detección de bucles cerrados

Como se ha dicho, la digitalización de grandes entornos 3D, con o sin oclusión, necesitan múltiples scans los cuales deberán ser registrados en un único mapa global. Supongamos el caso de un robot que se desplaza a lo largo de una trayectoria y que se detiene en $n + 1$ posiciones $V_0...V_n$ desde las que toma scans 3D. Un método directo para alinear los múltiples scans 3D es aplicar el

algoritmo ICP entre cada scan en la posición V_i y el scan de la posición previa V_{i-1}.

El emparejamiento ICP mejorará la estimación de las posiciones, sin embargo los errores en los registros de los scans se irán acumulando. Una forma de limitar este error usada el algoritmos de SLAM es la detección de bucles cerrados. La existencia de estos bucles hacen posible que los algoritmos de SLAM puedan deformar la trayectoria inicialmente estimada para, así, poder construir mapas consistentes globalmente.

Detectar que un robot se encuentra en una localización visitada previamente, puede ser muy complejo cuando se trabaja con scans 3D. Sin embargo, si se cuenta con una estimación odométrica con una cierta fiabilidad y con el algoritmo ICP, se puede detectar el cierre de un bucle a partir de la posición de los scan.

Nüchter et al. establecen que se ha detectado un bucle si dos posiciones distan menos de una cierta distancia (p.e. 5 metros) [16]. En dicho caso, se supone que existe una gran zona de solape entre ambos scans. Sin embargo, este criterio puede fallar si el láser 3D realiza su barrido con un haz muy cerrado. Sería preciso, entonces, determinar que, además de la distancia cercana, ambos scans deberán tener un número mínimo de puntos emparejados.

Además del método explícito comentado en los párrafos anteriores, el método GraphSLAM [17] utiliza un grafo de posiciones conectables. El método usado por Núchter en [4] es una extensión a seis grados de libertad del método de Lu y Millios [13]. La detección de bucles se realiza mientras se va procesando los scans mediante el ICP. Una vez detectado un bucle, se emplea un algoritmo de optimización de grafos 6D para la relajación global.

3.4. Método de dispersión global (global relaxation) basado en un grafo

Volvamos al caso del robot que se mueve siguiendo una trayectoria que pasa por $n+1$ posiciones $V_0, ..., V_n$ desde las que el robot se para y toma un scan del entorno. Si se emparejan dos scans hechos desde diferentes posiciones, se construye un conjunto de relaciones entre las posiciones. En el grafo resultante, cada nodo representa una posición y las aristas relaciones de vecindad entre las posiciones.

Dada una red como la anterior con $n+1$ nodos $X_0, ..., X_n$ y las aristas dirigidas $D_{i,j}$, se quiere estimar de forma óptima todas las poses para construir un mapa consistente del entorno. Se asume, por simplicidad, que la ecuación de medidas es lineal [14]:

$$D_{i,j} = X_i - X_j \tag{3.4}$$

La observación $\overline{D}_{i,j}$ de la auténtica diferencia se modela como $\overline{D}_{i,j} = D_{i,j} + \Delta D_{i,j}$, donde $\Delta D_{i,j}$

es un error con distribución de Gauss de media cero y matriz de covarianza $C_{i,j}$ que se asume conocida.

La posición óptima X_i se aproxima mediante estimación de máxima verosimilitud. Con la asunción de que todos los errores en las observaciones son Gaussianos y distribuidos independientemente, maximizar la probabilidad de todos los $D_{i,j}$, dadas sus observaciones $\overline{D}_{i,j}$, es equivalente a minimizar la siguiente distancia de Mahalanobis:

$$W = \sum_{(i,j)} (D_{i,j} - \overline{D}_{i,j})^T C_{i,j}^{-1} (D_{i,j} - \overline{D}_{i,j}) \tag{3.5}$$

3.5. Software de SLAM 6D

El software usado en este trabajo es el «6D Simultaneous Localization and Mapping»[2] y es mantenido por Andreas Nüchter y Kai Lingemann, de la Universität Onsnabrück. El software usado sirve para registrar nubes de puntos 3D en un sistema de coordenadas común. El software anterior es acompañado también de un visor para mostrar el escenario. Durante el registro de los scans 3D, se pueden usar diferentes algoritmos de minimización ICP, así como distintos métodos de dispersión global, siendo el objetivo final generar una escena completa y globalmente consistente.

3.5.1. Parámetros del SLAM 6D

El software dispone de una serie de parámetros que permiten controlar el comportamiento del algoritmo. Describimos a continuación los más significativos:

-a NR : Este parámetro permite seleccionar el método de minimización para el algoritmo ICP. Según el valor de NR podemos tener:

1. Aproximación de ángulo pequeño.
2. Método del cuaternión unitario de Horn.
3. Descomposición en valores singulares de Arun et al.
4. Matrices ortonormales por Horn et al.
5. Aproximación HELIX por Hofer y Potmann.

-c NR : Especifica la distancia máxima (NR unidades) a la que deben encontrarse dos posiciones para que se las considere el cierre de un bucle.

-d NR : Determina la distancia máxima (NR unidades) entre puntos para el emparejamiento mediante ICP.

[2] http://slam6d.sourceforge.net

-D NR : Determina la distancia máxima (NR unidades) entre puntos para el emparejamiento mediante SLAM.

-G NR : Este parámetro permite seleccionar el método de minimización para el algoritmo de SLAM. Según el valor de NR podemos tener:

1. Ninguna técnica de dispersión global.
2. Extensión de Lu & Milios mediante ángulos de Euler por Borrmann et al.
3. Extensión de Lu & Milios usando cuaterniones unitarios.
4. Aproximación HELIX por Hofer y Pottmann.
5. Aproximación de ángulo pequeño.

-i NR : Especifica el número máximo de iteraciones ICP.

-I NR : Especifica el número máximo de iteraciones SLAM.

-l : Fija el tamaño mínimo, en número de posiciones, que debe tener un bucle.

-m NR : Descarta todos los puntos a una distancia mayor que NR unidades.

-M NR : Descarta todos los puntos a una distancia menor que NR unidades.

-n FILE : especifica el fichero que contiene la estructura del grafo para el SLAM.

Una descripción más detallada del programa se puede encontrar en [18]. El uso de estos parámetros es heurístico [19]. Los autores indican en algunos ejemplos distintas configuraciones de parámetros para distintos entornos. Los distintos métodos de minimización no tienen un impacto significativo en la precisión del resultado. Sin embargo, la elección de la distancia máxima para el emparejamiento mediante ICP y mediante SLAM, así como la distancia al origen de los puntos que intervendrán en el algoritmo, sí que tienen un fuerte impacto en el resultado del mismo. También es muy importante determinar cuándo se considerará que se ha cerrado un bucle.

El seleccionar un valor pequeño de distancia para permitir el emparejamiento de puntos (parámetros -d y -D), o trabajar solamente con puntos cercanos al origen, hará que el error en el emparejamiento sea pequeño. Sin embargo, los puntos que disten mucho del sensor y que, por tanto, debido a la desviación en la orientación angular del robot, se hallen lejos de su correspondiente pareja no serán emparejados. Este emparejamiento no efectuado habría supuesto una corrección significativa en la orientación. Sin embargo, por el contrario, permitir una distancia muy grande para emparejar puntos aumenta la probabilidad de emparejar puntos que no se corresponden a la misma estructura del entorno real.

La distancia entre posiciones, para considerar que se ha cerrado un bucle, debe ser elegido de acuerdo al entorno y a las características del barrido del scan láser. Un entorno con muchos objetos que producen oclusión puede conllevar que dos scans, incluso a distancias relativamente cortas, no

se solapen suficientemente. En este caso se debería limitar la condición de cierre de bucle sólo a distancias muy cortas entre scans. Asimismo, si el haz del sensor láser es muy estrecho, o su área de barrido muy corta, debe preverse que pequeñas diferencias en la orientación del robot pueden dar como resultados scans que no se solapen. En estos casos, en vez de utilizar la distancia entre posiciones como condición de cierre de bucle, sería preferible usar directamente como condición un número mínimo de puntos de solapes entre los scans.

Capítulo 4

Etiquetado semántico de elementos del entorno

Reconocer qué es lo que se encuentra en el entorno es importante para poder desarrollar agentes realmente autónomos. Por esto, es importante generar mapas semánticos. Según Nüchter [4]:

> Un mapa 3D semántico para un robot móvil es un mapa métrico, el cual, además de la información geométrica de los datos 3D, contiene asignaciones entre esos puntos y estructuras o clases de objetos conocidos

Los distintos scans adquiridos, una vez corregida su posición con la librería SLAM 6D, se registran en un mapa de superficies multinivel (MSM) [5]. En este trabajo se automatiza la detección de las siguientes estructuras, suelos, techos, mesas, paredes y puertas, a partir del MSM. Las cinco primeras estructuras se corresponden con estructuras planas. Para detectarlas se aplicará el algoritmo «Efficient Ransac para Mapas de Superficie Multinivel»(eRMSM) [11] y, posteriormente, se clasificarán los planos obtenidos de acuerdo a su orientación y altitud. La localización de posibles puertas pretende localizar zonas de paso entre distintos espacios del entorno. La identificación de tales puertas requerirá un análisis de los planos para encontrar huecos que cumplan una serie de condiciones, como se explicará más adelante.

4.1. Mapas de superficie multinivel

En este trabajo se ha empleado como sistema de representación 3D un mapa de superficies multinivel. En [6] se propone una extensión de los mapas de elevación hacia las múltiples superficies, los llamados mapas de superficies multinivel, que ofrece la oportunidad de modelar el entorno con más de un nivel transitable. El conocimiento acerca de las superficies horizontales es adecuado para dar soporte al análisis de transitabilidad y planificación de trayectos. Sin embargo, únicamente proporciona un débil soporte a la localización de vehículos o al registro de diferentes mapas. Modelar únicamente las superficies significa que las estructuras verticales, percibidas frecuentemente por vehículos anclados en el suelo, no pueden ser usadas para apoyar la localización y el registro. Para solventar esta dificultad, en [6] se evita este problema representando los intervalos correspondientes

Figura 4.1: Ejemplo de entorno con múltiples superficies: el suelo, el asiento de las sillas y la mesa

a objetos verticales del entorno. Esta solución presenta la ventaja de poder almacenarse de forma compacta y al mismo tiempo puede ser usada para dar soporte al problema de la asociación de datos durante la alineación de mapas.

Un mapa de superficies multinivel consiste en una rejilla de dos dimensiones de tamaño variable en la que cada celda, $c_{i,j}$, almacena una lista de parches de superficie, $p_{ij}^1, \cdots, p_{ij}^k$. Un parche de superficie se representa como la media y la varianza de la alturas medidas en la posición de la celda $c_{i,j}$ en el mapa y refleja la posibilidad de atravesar el entorno 3D a la altura dada por la media μ_{ij}^k, mientras que la incertidumbre de esta altura se representa mediante la varianza σ_{ij}^k. En [6] se asume que el error en la altura sigue una distribución gausiana por lo que usan indistintamente «Gausiano en una celda» y «parche en una celda».

Además de la media y la varianza, se almacena un valor de profundidad en cada parche de superficie. Este valor de superficie refleja el hecho de que un parche de superficie puede estar en la cima de un objeto vertical, como pudiera ser un edificio, un puente o una rampa. En esos casos, la profundidad se define como la diferencia de la altura h_{ij}^k de la superficie y la altura $h_{ij}'^k$ de la medida más baja perteneciente al objeto vertical. Un objeto plano, como puede ser el suelo, tendrá profundidad 0.

Se puede generar un mapa de superficies multinivel de dos formas distintas, a partir de un conjunto de mediciones 3D, como puede ser una nube de puntos con sus varianzas, o bien uniendo dos mapas de superficies multinivel en uno solo. Ambas formas son equivalentes, esto es, si se crea el mapa m_1 a partir de la nube de puntos C_1 y el mapa m_2 a partir de la nube C_2, entonces el mapa m_3 que resulte de la unión de los dos mapas anteriores es idéntico al mapa generado a partir de la nube $C_3 = C_1 \cup C_2$. Dado una nube de puntos C con varianzas $\sigma_1, \cdots, \sigma_n$ el mapa correspondiente se crea de la siguiente forma:

- Cada celda del mapa $c_{i,j}$ recogerá todos los puntos $p = (x, y, z)$ tal que $s \cdot i \leq x \leq s \cdot (i + 1)$

y $s \cdot j \leq y \leq s \cdot (j+1)$ donde s denota el tamaño del lado de una celda del mapa.

- En cada celda se calcula un conjunto de intervalos en altura a partir de la altitud de los puntos. Si dos alturas distan menos que un determinado valor de «hueco», de tamaño γ, se las considera pertenecientes al mismo intervalo. Este tamaño de «hueco» se elige en [6] de forma que el sistema completo (robot más sensores) pueda pasar a través del hueco.

- Los intervalos se clasifican como estructuras horizontales o verticales. La diferencia entre ambas estructuras radica en la altura del intervalo. Si esta altura es mayor que un valor de «grosor» τ se considera al intervalo vertical, en caso contrario se considera horizontal.

- En los intervalos verticales se almacenan la media y la varianza de la medida más alta junto con la altura. El valor de la altura se usará también para emparejar juntos dos mapas. Cuando se desea emparejar dos mapas adquiridos desde distintas posiciones, un algoritmo muy usado es el ICP originalmente desarrollado por Besl y Mckay [15] o alguna de sus múltiples variantes. El método, ideado para alinear nubes de puntos, se adapta mal a su aplicación sobre gausianos. Por eso, en [6] se usa la altura de los gausianos para clasificarlos como estructuras verticales o no y así aplicar el método ICP por separado con los distintos tipos de gausianos.

- En los intervalos horizontales se calcula la media y la varianza de todas las medidas. Para hacer esto se utiliza un filtro de Kalman aplicada a todas las medidas. En estos intervalos la profundidad se establece como cero.

Una vez calculadas las medias, varianzas y alturas de cada parche de superficie, la nube de puntos se borra y todo cálculo posterior se lleva a cabo únicamente con los datos del mapa. Esto reducirá notablemente la cantidad de memoria necesaria.

El procedimiento que se sigue cuando se añade una nueva medida se muestra en la figura 4.2 expresado en pseudocódigo. Cuando se añade una nueva medida $z = (p, \sigma)$, siendo p las coordenadas del punto y σ su varianza, a un mapa de superficies multinivel, lo primero que se necesita saber es si la medida es parte de algún parche de superfice que ya está representado en el mapa o si corresponde a alguno nuevo. Con este fin, primero se determina cuál es la celda c_{ij} en la cual la nueva medida cae en función de las coordenadas (p_x, p_y). Después se busca en c_{ij} el Gausiano cuya media esté más cerca de la altura de z. Si el Gausiano está lo suficientemente cerca, se adapta con esta nueva medida usando de nuevo la regla de adaptación de Kalman. En [6] se establece que un Gausiano está suficientemente próximo a la medida z si la altura de z dista menos de 3σ del Gausiano. Vemos las tres situaciones posibles en la figura 4.3

Si z está lejos del Gausiano más cercano, puede aún pertenecer a un objeto vertical. Esto se puede determinar comprobando si z está dentro de la altura ocupada por uno de los Gausianos en cuyo caso z es descartado. Finalmente, si z no perteneciera a ningún Gausiano se crea con él un nuevo Gausiano.

```
añadir_nueva_medida
entrada:
px, py, pz - coordenadas espaciales de la medida
σ - varianza de la medida
s - longitud de la celda
i := entero (px/s)
j := entero(py/s)
G := gausiano_más_próximo(cij,pz)
si G es nulo entonces
          añadir_gausiano(cij,px,py,pz,σ)
          retornar
si no
          si distancia(G.z,pz)<3·σ entonces
                    actualizar_gausiano(G,px,py,pz,σ)
                    retornar
          si no
                    si gausiano_coincidente(cij,pz) es nulo
entonces

          añadir_nuevo_gausiano(cij,px,py,pz,σ)
                              retornar
                    fin si
          fin si
fin si
```

Figura 4.2: Algoritmo para añadir una nueva medida a un mapa de superficies multinivel.

Además de la distinción entre estructuras horizontales y verticales, las estructuras horizontales también pueden ser clasificadas como transitables o no transitables. El concepto refleja el hecho de que ciertas estructuras horizontales no pueden ser alcanzadas por el robot debido a que se encuentran aisladas. En [6] se determina que un parche de superficie sólo puede ser transitable si al menos 5 de las 8 celdas vecinas existen y si la diferencia de alturas entre el parche y cada uno de sus vecinos es menor de 10 cm. Estos valores pueden ajustarse según interese en distintos escenarios.

4.2. Detección y etiquetado de planos

El algoritmo eRMSM es una variante del algoritmo *eRansac* [9]. El algoritmo *eRansac* opera sobre nubes de puntos. El eRMSM trabaja sobre los Gaussianos constituyentes de los mapas multinivel.

El algoritmo opera sobre todo el mapa de superficies multinivel. Esto evita tener que almacenar todas las coordenadas que se van generando por parte del sensor láser, lo que multiplicaría varias veces las necesidades de memoria. Por otra parte, al ser más compacta la representación sobre la que opera el algoritmo, el proceso de descubrimiento de planos en el entorno se acelera en gran medida. Además, se ha introducido una modificación en el algoritmo eRANSAC lo que permite una convergencia más rápida en el descubrimiento de planos.

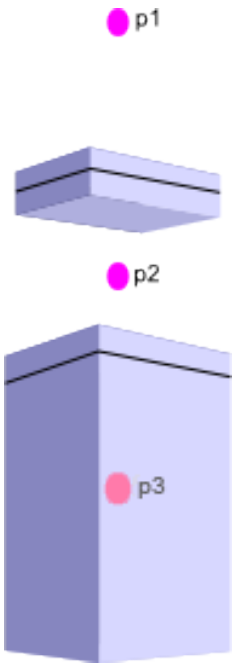

Figura 4.3: Distintas situaciones a la hora de insertar una nueva medida en un mapa multinivel. El punto p1 está alejado de cualquier gausiano por lo que se creará uno nuevo para él. El punto p2 se encuentra próximo a una estructura horizontal, por lo que procederá a actualizar la media y la varianza de esta mediante filtros de Kalman. Finalmente, el punto p3 está en el interior de una estructura vertical por lo que no es necesario realizar ninguna actualización.

Extraer las superficies planas presentes en el mapa es un proceso de ensayo y prueba. De forma iterativa se van generando planos candidatos. Estos planos candidatos se crean a partir de los bloques presentes en el mapa multinivel. Antes de iniciar el bucle de generación y testeo de planos candidatos, hay que calcular los vectores normales a cada bloque. Estas normales nos permiten reducir el número de bloques necesarios para generar cada plano candidato y para descartar de entrada candidatos que no llevarán a nada. Este vector normal se estima mediante el análisis de componentes principales (ACP). Este método, ideado para encontrar la dimensión o dimensiones en donde se encuentra la mayor variación de un conjunto de datos, también puede ser usado para estimar la normal a una superficie que se desconoce pero de la que se poseen algunos puntos. Para cada bloque b_i se aplica el ACP al conjunto formado por él mismo y los bloques b_j próximos. El método se aplica de la siguiente forma:

1. Se genera la matriz de covarianza de los elementos b_i y b_j.

2. Se calculan los autovectores, v_k, y autovalores, a_k, de la matriz de covarianza.

3. El autovector correspondiente al menor autovalor será justamente la estimación del vector normal a la superficie hipotética que uniría todos los puntos, en nuestro caso, bloques.

Los pasos que seguimos en nuestro algoritmo de detección de planos son:

1. Se selecciona al azar un bloque del mundo. Este bloque puede ser de tipo superficial, altura igual a cero, o puede ser de tipo vertical, con altura distinta de cero. Según el tipo de este primer bloque el resto de la iteración opera sólo con bloques del mismo tipo: superficiales o verticales. Esto está justificado por el hecho de que las superficies planas están formadas por bloques del mismo tipo. Una pared, por ejemplo, está toda ella formada por bloques verticales. Sin embargo, el tablero de una mesa está formado por bloques superficiales por completo.

2. Se seleccionan al azar dos bloques en la vecindad del bloque seleccionado en el primer paso. Los vectores normales a estos dos bloques no deben diferir del vector normal del bloque del punto anterior más de *MAX_ANGLE* grados, siendo éste un parámetro del sistema. El radio de la vecindad en que se seleccionan los bloques vendrá determinado por el parámetro del sistema *RADIUS*. Este radio se especifica en número de celdas. Por tanto, el alcance real de este radio, expresado en milímetros será $RADIUS \times CELL\_SIZE$. Para determinar la distancia entre dos bloques se usa la distancia de Chebyshev[1] por ser menos costosa computacionalmente. Este radio determinará el comportamiento del algoritmo. Cuanto más pequeño sea, más probabilidad hay de que los bloques seleccionados pertenezcan al mismo plano. Sin embargo, cuanto más próximos sean los bloques, el plano candidato que se genere tendrá más probabilidades de estar influido por anomalías locales como puede ser una irregularidad en una pared o suelo (ver fig. 4.4-a).

[1] $d_\infty(x, y) = max_{j=1\cdots J} |x_j - y_j|$

3. Se genera un plano a partir de los tres bloques seleccionados en los puntos anteriores si se cumplen las condiciones ya explicadas. El plano se modela mediante un punto y un vector normal. Como punto se elige el baricentro de los tres bloques y como vector normal se elige el vector promedio de los vectores normales a los tres bloques. Esta es una modificación respecto a *efficient-Ransac* en donde la normal al plano se tomaba como la normal al primer punto seleccionado. Entendemos, sin embargo, que utilizar como normal el promedio de los tres bloques seleccionados minimiza el error que un único elemento aislado de una superficie no perfecta puede tener. La velocidad de convergencia aumenta al introducir este cambio como se muestra en la figura 4.4-b y c.

4. Se asigna una puntuación al plano candidato. Esta puntuación es el número de bloques del mundo que se consideran coincidentes con el plano candidato. El concepto coincidente implica que su distancia al plano es menor que el parámetro *EPSILON* y su normal no difiere más de *MAX_ANGLE* grados de la normal del plano.

5. Entre todos los planos candidatos que poseamos, se elige al de mayor puntuación y sólo se admite como válido si la probabilidad de elegir al azar dicho plano es inferior a $(1 - PROBABILITY)$, siendo $PROBABILITY$ un parámetro del sistema. En el algoritmo *eRansac*, el cálculo de esta probabilidad se hace tomando en cuenta todas las combinaciones entre todos los puntos de la nube. Esto, siendo robusto matemáticamente, sin embargo no utiliza el conocimiento acerca de cómo se ha generado el plano candidato. Como consecuencia, el *eRansac* exige que se genere un número enorme de planos candidatos antes de aceptar a uno de ellos como válido.

6. En caso de haber encontrado un plano que supere el test de probabilidad, se almacena en una lista y se eliminan los planos candidatos que tuvieran bloques coincidentes con el plano ganador. Los bloques coincidentes con el plano ganador ya no vuelven a ser usados para generar nuevos planos ni para puntuar a los futuros candidatos.

Para estimar los factores anteriores, consideremos una nube de puntos P de tamaño N y una forma ψ que consta de n puntos. Sea k el tamaño del conjunto mínimo necesario para definir dicha forma candidata. Debido a que cualesquiera k puntos de la forma llevarían a generar también esa forma candidata, entonces la probabilidad de detectar ψ en un único paso es:

$$P(n) = \binom{n}{k} / \binom{N}{k} \approx \left(\frac{n}{N}\right)^k \tag{4.1}$$

La probabilidad de tener éxito en una detección, $P(n, s)$, después de que s formas candidatas hayan sido generadas es igual al complementario de s fallos consecutivos:

$$P(n, s) = 1 - (1 - P(n))^s \tag{4.2}$$

Así, finalmente, de la ecuación anterior, resolviendo s, obtenemos el número mínimo de candidatas que hay que generar, T, para detectar formas de un tamaño n con una probabilidad $P(n, T) \geq p_t$:

$$T \geq \frac{ln(1-p_t)}{ln(1-P(n))} \tag{4.3}$$

Ilustraremos esto con un ejemplo. Supongamos un escenario formado por un tablero de una mesa y dos paredes haciendo esquina con la mesa. Todo el entorno está constituido por tres planos ortogonales. Sea 3000 el número de puntos que forman el mapa y cada uno de los tres planos está representado por 1000 puntos en el mapa. Generamos un plano candidato, P_c, que coincide con una de las superficies de nuestro escenario, supongamos que la mesa. Ese plano candidato recibirá una puntuación igual a la tercera parte de los puntos totales. Calculemos ahora cuántos planos candidatos más hacen falta generar antes de aceptar al plano de la mesa como válido con una probabilidad igual o superior a 0,99. Recordemos que el número de puntos k usado para generar un plano candidato es igual a 3. Entonces, de la ecuación 4.1 tenemos que la probabilidad de haber encontrado a nuestro plano candidato en un único paso es igual a:

$$P(n) = \left(\frac{1000}{3000}\right)^3 \approx 0{,}037$$

Obtenemos finalmente de la ecuación 4.3 el número de planos candidatos necesarios para aceptar a P_c con una probabilidad igual o superior a 0,99:

$$T \geq \frac{ln(1-0{,}99)}{ln(1-0{,}037)} \approx 122$$

Es decir, aunque desde el comienzo tenemos un plano candidato que se corresponde con un plano del entorno, ha sido necesario generar otros 121 planos candidatos antes de tomar a P_c como válido. Sin duda el método es matemáticamente preciso, pero se puede acelerar si tenemos en cuenta que los planos candidatos son generados a partir sólo de puntos, bloques en nuestro caso, que tienen unas normales que necesariamente tienen que forman un ángulo no mayor de MAX_ANGLE grados. Por tanto, ya que los planos candidatos no se pueden generar con k puntos cualesquiera del mapa, sino solo con k puntos con normales similares, para evaluar la probabilidad de haber generado un plano candidato válido podemos considerar sólo los puntos con una normal similar a la del plano candidato. Esto acelera en gran medida el número de iteraciones necesarias para descubrir planos válidos. En el ejemplo expuesto anteriormente, la probabilidad de haber encontrado a nuestro plano candidato en un único paso sería igual a:

$$P(n) = \left(\frac{1000}{1000}\right)^3 = 1$$

Siendo por tanto el número mínimo de planos candidatos necesarios para aceptar a P_c igual a:

$$T \geq \frac{ln(1-0{,}99)}{ln(1-1)} = 0$$

En un escenario real hay numerosos planos con una orientación única en el entorno. Esos planos podrán descubrirse en un único paso, una vez generado el plano candidato que los caracterice. Otros planos podrán no tener orientaciones únicas, por ejemplo las paredes paralelas entre si, o el techo

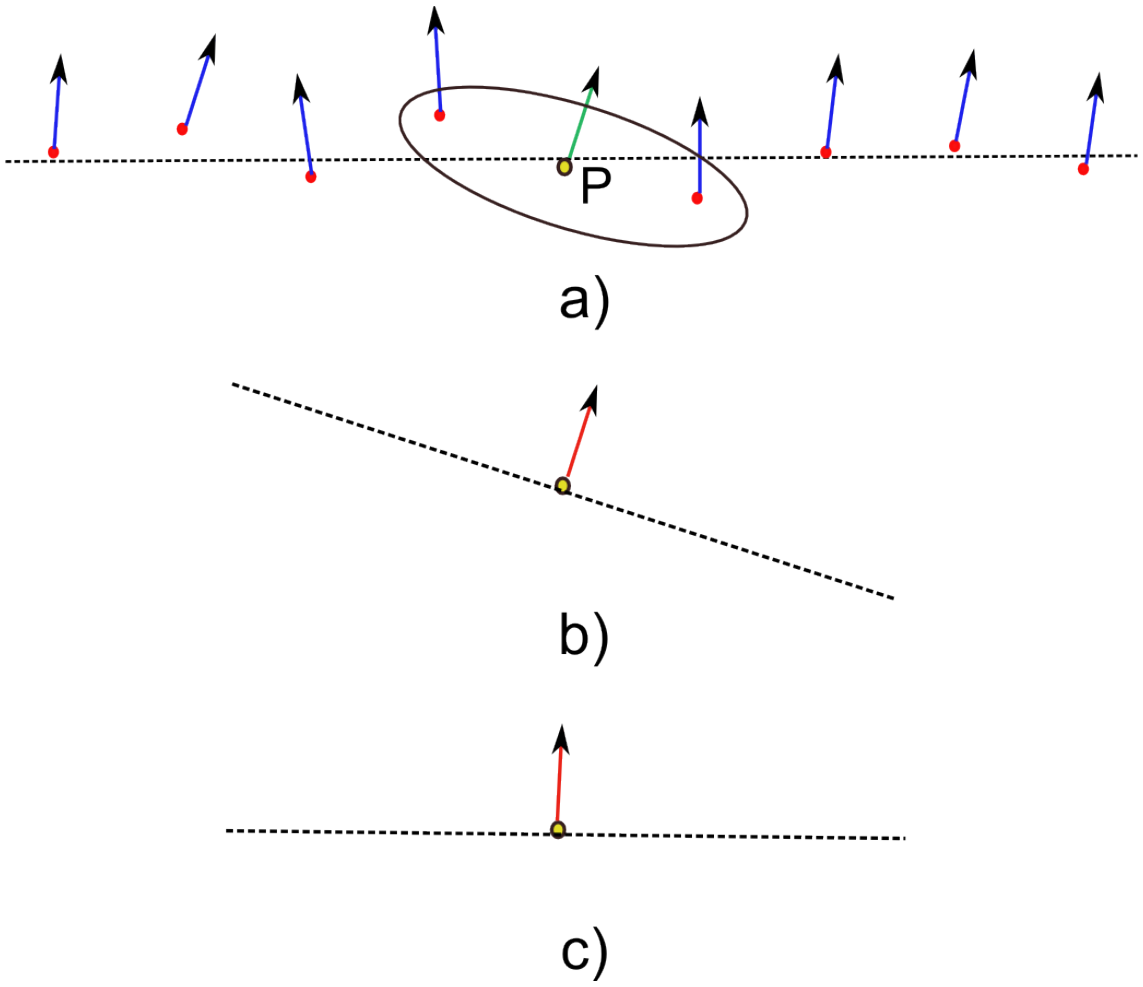

Figura 4.4: a) Elegir una vecindad muy corta reporta el beneficio de aumentar las probabilidades de que los puntos seleccionados pertenezcan a un mismo plano. Sin embargo, la contrapartida es que la determinación de las normales a cada punto se ve muy afectadas por las pequeñas irregularidades del objeto o por la incertidumbre inherente al aparato de medición. b) Tomar como vector normal de los planos candidatos la normal del primer punto P seleccionado conlleva que se generen numerosos planos espúreos si la normal al punto P se desvía de la normal al plano verdadero. c) Tomar como normal al plano candidato el promedio de las normales a los puntos seleccionados aumenta las probabilidad de que el plano candidato sea aceptado

y el suelo, pero en cualquier caso, esta modificación reduce notablemente el tiempo necesario para validad dichos planos.

A cada plano encontrado se le asigna un color que servirá posteriormente para etiquetar cada uno de los bloques coincidentes con él y para representarlos gráficamente.

4.2.1. Etiquetado de los planos

Una vez extraídos los planos, estos deben ser etiquetados. Los planos serán clasificados en los siguientes tipos: suelos, paredes, techos y mesas. A continuación se explica el criterio seguido para el etiquetado de cada categoría.

Suelo. Se considera suelo a un plano que cumpla dos condiciones. Primera, que el valor absoluto de la componente N_z del vector $\vec{N}$ unitario normal al plano, sea mayor que 0,9. Segundo, que los puntos del plano, dentro de la escena, disten menos de CELL_SIZE mm. del plano XY.

Techo. Este es un plano también paralelo, o prácticamente paralelo, al plano XY pero, entre todos los que cumplen esta condición, es el que mayor altitud tiene sobre el suelo.

Mesas. Son aquellos planos paralelos al plano XY que se encuentran en altura entre el suelo y el techo.

Paredes. Son los planos perpendiculares al plano XY. Se considera condición suficiente que la componente N_z del vector unitario normal al plano sea menor que 0,2.

Estos planos así etiquetados se recogen en el fichero de salida «ssms.log».

4.3. Localización de puertas

La localización de puertas se hace analizando los planos previamente etiquetados como paredes. A la hora de establecer un algoritmo para la detección de puertas, hay que tener en cuenta que la sección del mapa correspondiente a la pared con la que se trabaja dista, normalmente, mucho de ser una superficie perfectamente rectangular.

El algoritmo desarrollado para la detección de puertas trabaja sobre las paredes detectadas. Para cada pared, se crean dos vectores L y U de igual tamaño. El tamaño S de los vectores se obtiene dividiendo la longitud total de la pared, L, por el tamaño de celda del MSM, $CELL\_SIZE$. El algoritmo procede de la siguiente forma:

1. Se almacena en cada celda del vector L la altura inferior de las estructuras del mapa que se hallen sobre la vertical de dicha celda.

2. Se aplica un filtro mediana al vector L para suavizar y eliminar errores en el plano de pared con el que se trabaja.

3. Se umbraliza el vector L sobre el vector U. En U se almacenará un -1 si la celda correspondiente de L es menor que 1500,0, un 1 si es mayor que 1500,0 y un 0 en caso de que la celda de L esté vacía.

4. Se busca en U la siguiente cadena, dada aquí en forma de expresión regular: $(-1)(-1)[-1]*(1)(1)[1]*(-1)[-1]*$. Esta cadena en U representa dos zonas de pared baja (por debajo de los 1500 mm.) con una zona de pared alta (por encima de los 1500 mm) en medio, lo que se propone como posible puerta.

El algoritmo puede producir falsos positivos debido a que el plano que forma la pared no esté completamente definido por oclusiones de otros objetos y por no haber sido totalmente barrido por el láser. En cualquier caso, el interés de la detección de las posibles puertas consiste en señalar al sistema posible zonas de interés y atraer su atención para una exploración más detallada.

Capítulo 5

Resultados

Mostraremos en este capítulo los resultados obtenidos tras aplicar cada paso del proceso aquí implementado a los escenarios elegidos como prueba. Tras la adquisición de scans se muestra como sería el MSM resultante si se creara exclusivamente a partir de la información odométrica. Se hará evidente que se acumula un gran error. Tras la corrección efectuada mediante el SLAM 6D se observa que se ha conseguido una gran corrección consiguiéndose construir un mapa coherente. Finalmente, tras el paso de etiquetado se mostrarán las estructuras que se ha conseguido identificar.

5.1. Resultados de la adquisición de datos

Como se comenta en la sección 2.2, los escenarios elegidos son un gran pasillo y un laboratorio. En el primer caso se tomaron 24 scans 3D, mientras que en el segundo se tomaron 8. Los scans 3D se realizaron haciendo un barrido entre los $-20°$ y los $20°$ de inclinación acimutal y usando todo el rango del haz láser ($270°$). La distancia media entre las posiciones en el laboratorio, según la odometría fue de 3627,85mm. En el caso del pasillo la distancia media según la odometría fue de $13054,59$mm. Las figuras 5.1 y 5.2 muestran una vista 3D de los mapas creados a partir de la odometría. El mapa del laboratorio está creado con un tamaño de celda de $20x20$mm. y el del pasillo con un tamaño de celda de $100x100$mm. Es evidente cómo la acumulación del error hace que el mapa construido no es coherente. En las figuras se señala dentro de un círculo algunas zonas donde es fácil observar la desviación producida.

5.2. Resultados del SLAM 6D

5.2.1. Creación de dos mapas independientes

La herramienta SLAM 6D ha demostrado ser un instrumento eficaz en la tarea de minimizar el error proporcionado por la odometría y conseguir registrar correctamente los múltiples scans en un mapa de superficies multinivel coherente. Se muestran en las figuras 5.3 y 5.4 los mapas de superficies multinivel creados a partir de las posiciones corregidas tras aplicar el algoritmo de

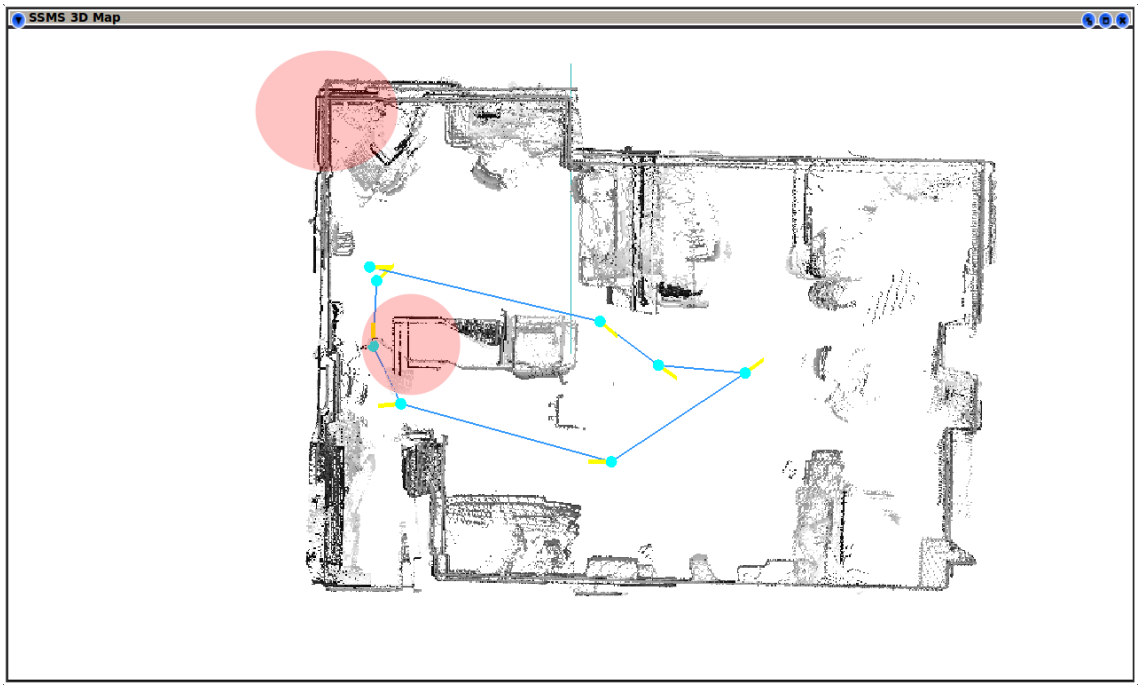

Figura 5.1: Vista aérea del mapa de superficies multinivel del laboratorio construido a partir de la información odométrica. En los círculos, zonas donde se aprecia una gran desviación entre los distintos scans. La línea marca la trayectoria seguida por el robot y los puntos la posición desde la que se tomó el scan

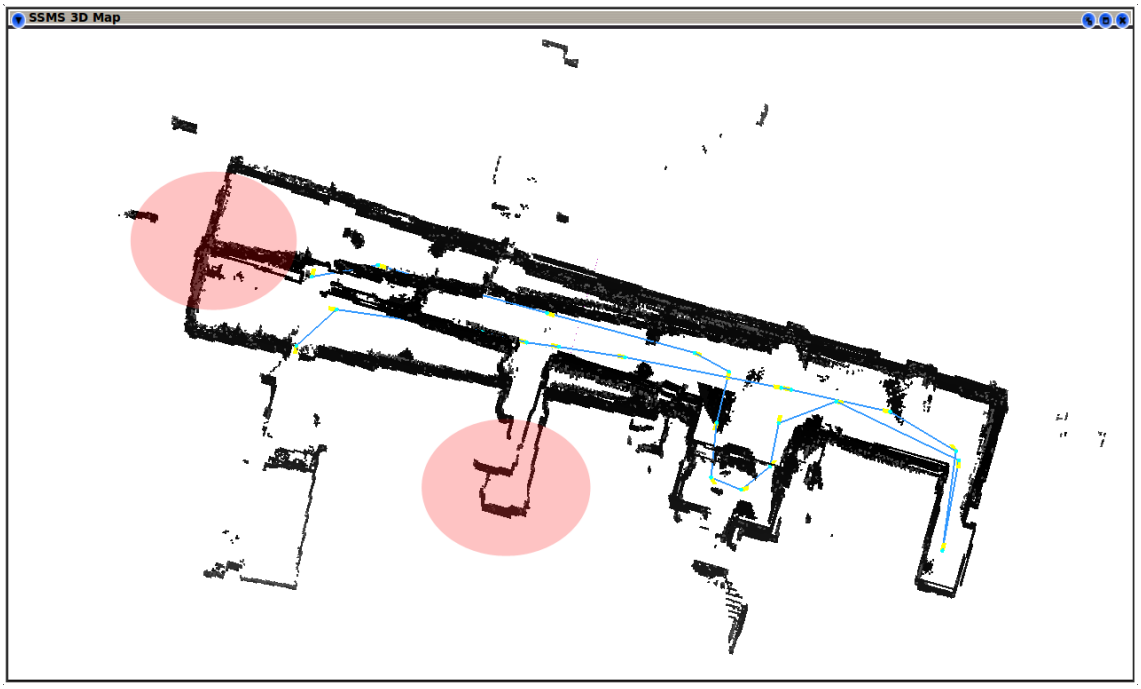

Figura 5.2: Vista aérea oblicua del mapa de superficies multinivel de un pasillo construido a partir de la información odométrica. En los círculos, zonas donde se aprecia una gran desviación entre los distintos scans. La línea marca la trayectoria seguida por el robot y los puntos la posición desde la que se tomó el scan

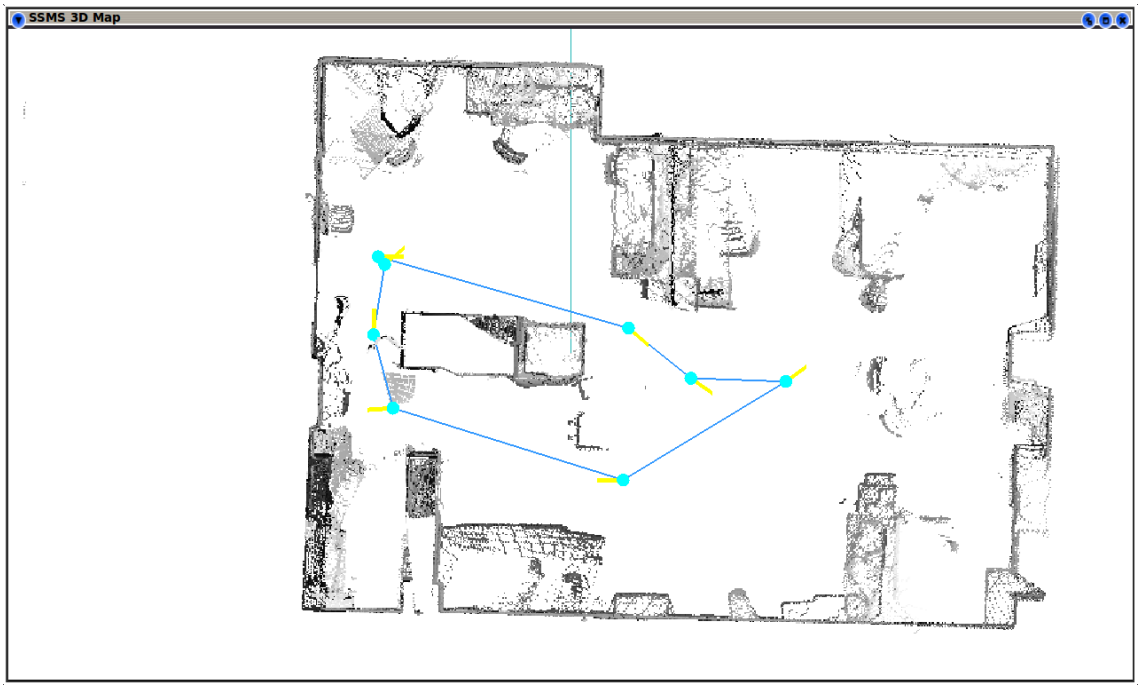

Figura 5.3: Vista aérea del mapa de superficies multinivel del laboratorio construido a partir de las posiciones corregidas por el algoritmo SLMA 6D. La línea marca la trayectoria seguida por el robot y los puntos la posición desde la que se tomó el scan

SLAM 6D. Igual que en el caso anterior, el mapa del laboratorio está creado con un tamaño de celda de $20x20$mm. y el del pasillo con un tamaño de celda de $100x100$mm. Si comparamos la zona marcada con un círculo en las figuras 5.1 y 5.2 con la región correspondiente en las figuras 5.3 y 5.4 observamos cómo se ha corregido de forma efectiva el error odométrico.

El algoritmo SLAM 6D ha sido aplicado al pasillo usando los siguientes parámetros:

- Distancia máxima para emparejar puntos durante el ICP: 500 mm.
- Distancia máxima para emparejar puntos durante el SLAM: 500 mm.
- Distancia mínima de un punto al origen para ser emparejable: 3000 mm.
- Número máximo de iteraciones en ICP y en SLAM: 50
- Distancia máxima para considerar que dos posiciones cierran un bucle: 5000 mm.
- Número de scans mínimo para considerar un cierre de bucle: 2

Se ha conseguido, tras la aplicación del SLAM 6D, un error medio de tan solo 11 mm. Al ser el pasillo un escenario de grandes dimensiones, hemos preferido trabajar sólo con puntos que disten más de 3m. del sensor (parámetro -M 3000). En dos scans distintos, los puntos correspondientes a

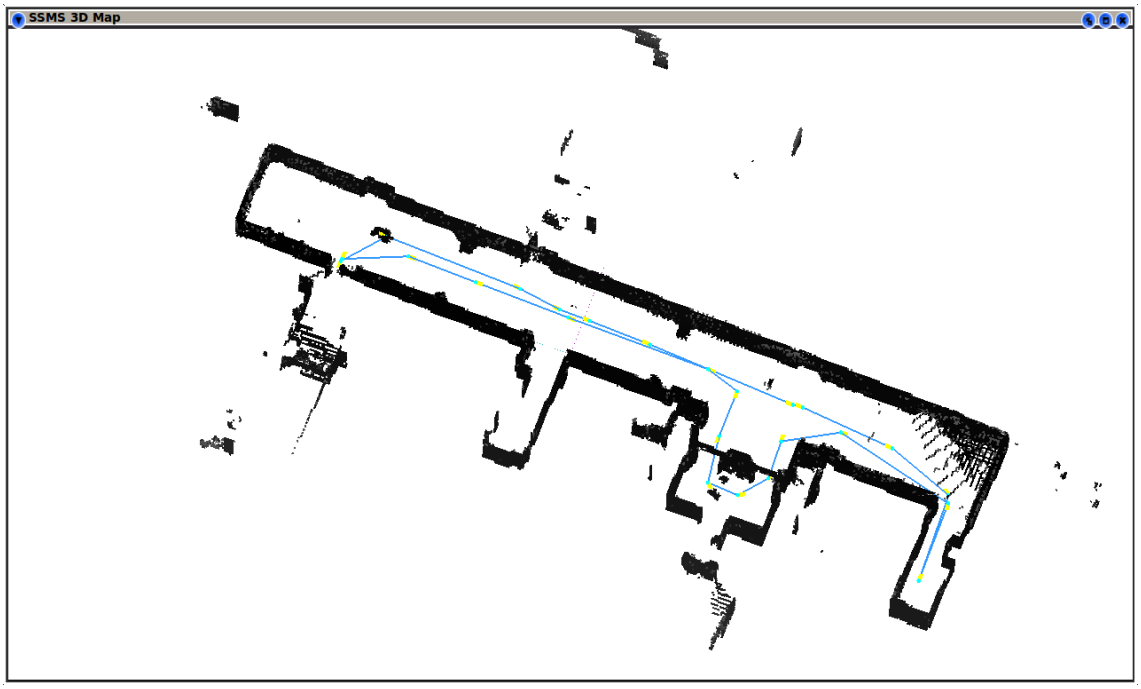

Figura 5.4: Vista aérea oblicua del mapa de superficies multinivel de un pasillo construido a partir de las posiciones corregidas por el algoritmo SLAM 6D. La línea marca la trayectoria seguida por el robot y los puntos la posición desde la que se tomó el scan

la misma localización espacial, debido a los errores odométricos en la estimación de la orientación y a su lejanía del sensor, distaran bastante entre sí, lo que forzará una gran corrección en la estimación odométrica. Por otra parte, esto nos obliga a permitar una mayor distancia entre pares de puntos para que estos sean emparejables (parámetro -d 500).

Vemos en las figuras 5.5 y 5.6 la localización espacial de las distintas posiciones desde las que se tomaron los scans 3D y el grafo que se proporcionó al algoritmo SLAM 6D. En este caso, se han establecido tres cierres de bucles. Entre las posiciones 12 y 14, entre la 3 y la 20 y entre la 0 y la 23.

En el caso del laboratorio, el algoritmo SLAM 6D ha sido aplicado usando los siguientes parámetros:

- Distancia máxima para emparejar puntos durante el ICP: 100 mm.
- Distancia máxima para emparejar puntos durante el SLAM: 200 mm.
- Número máximo de iteraciones en ICP y en SLAM: 50
- Distancia máxima para considerar que dos posiciones cierran un bucle: 5000 mm.
- Número de scans mínimo para considerar un cierre de bucle: 2

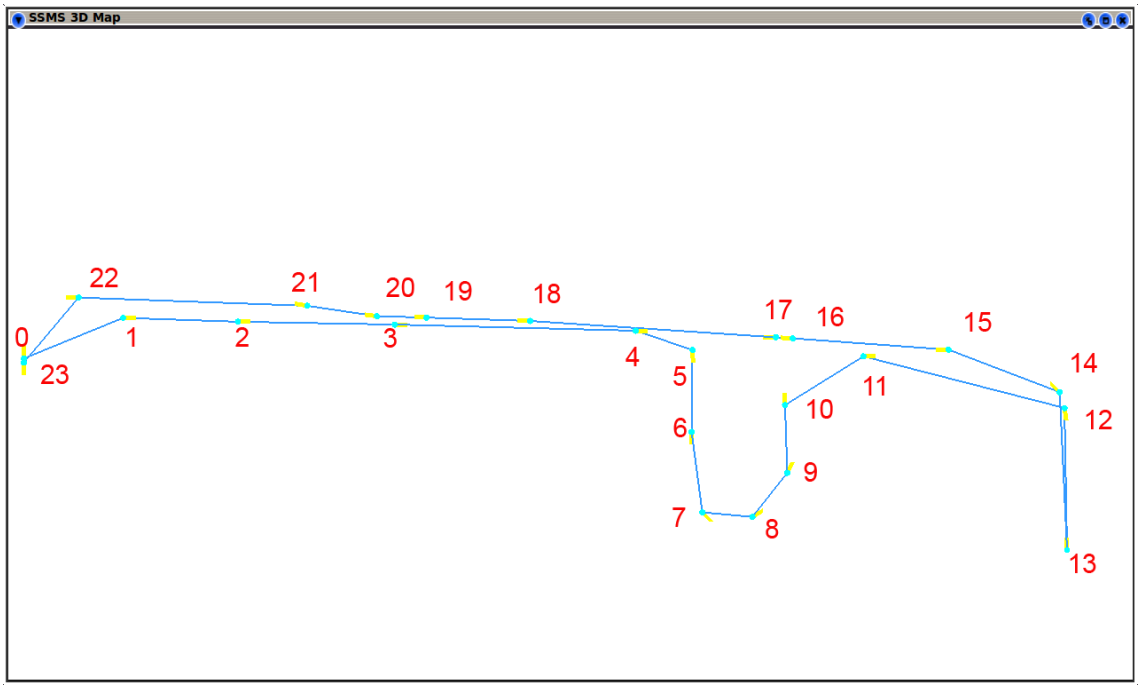

Figura 5.5: Localización de las distintas posiciones desde las que se tomaron los scans 3D en el pasillo. En la imagen, el eje X es el vertical con el sentido positivo hacia arriba y el Y es el horizontal con el sentido positivo hacia la izquierda

Se ha conseguido en esta ocasión, tras la aplicación del SLAM 6D, un error medio de apenas 3.65mm. Al ser el laboratorio un espacio más pequeño que el pasillo, hemos trabajado con todos los puntos, independientemente de su distancia al sensor. En esta ocasión la distancia máxima para emparejar puntos durante el ICP (parámetro -d 100) es menor que en el pasillo también debido a las menores dimensiones del laboratorio. La distancia máxima permitida para emparejar puntos durante el SLAM es mayor que durante el ICP (parámetro -D 200) debido a que, al distribuir el algoritmo de SLAM el error entre sucesivos scans se habrá acumulado el error y conviene trabajar con entornos mayores.

Vemos en las figuras 5.7 y 5.8 la localización espacial de las distintas posiciones desde las que se tomaron los scans 3D y el grafo que se proporcionó al algoritmo SLAM 6D.

Se ha observado, en los mapas construidos, una desviación respecto a la realidad. Si observamos una proyección del pasillo sobre el plano YZ (ver figura 5.9), observamos cómo el suelo y el techo van «cayendo» a medida que nos desplazamos a la derecha. Esto tiene dos causas. Por una parte, cada scan 3D se efectuó entre los $-20°$ y los $20°$. Esto motiva que el haz láser incida de forma bastante tangencial tanto al suelo como al techo. A esto se suma que, en el caso del suelo, una superficie brillante provoque rebotes en el haz o incluso medidas erróneas. Como resultado de lo anterior, tenemos una pobre reconstrucción del suelo y el techo. Como ejemplo, la figura 5.10 muestra el scan número 14 del pasillo. Vemos como el techo tiene una muy baja densidad de puntos

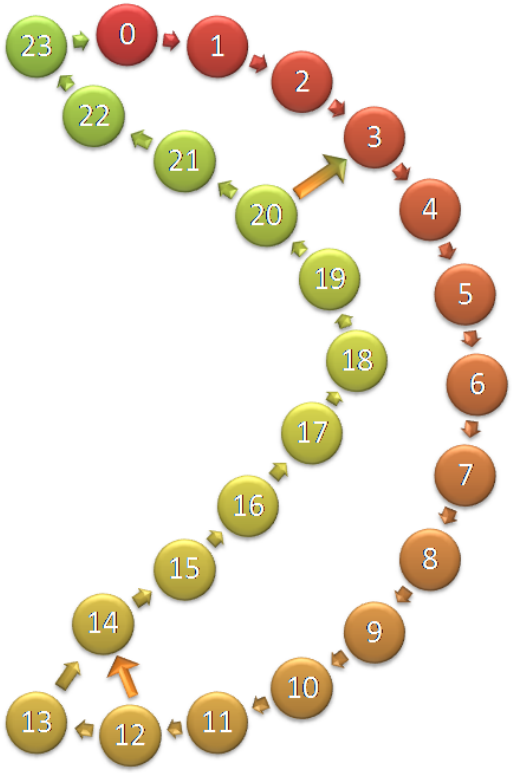

Figura 5.6: Grafo de restricciones entre las posiciones desde las que se tomaron los scans del pasillo

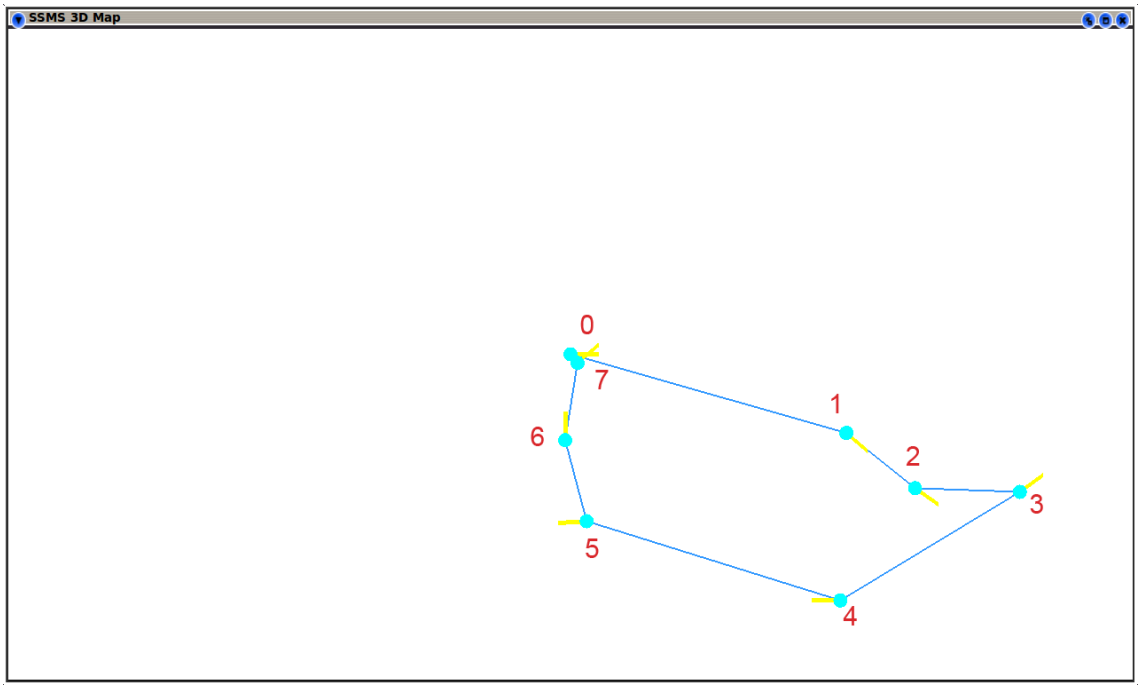

Figura 5.7: Localización de las distintas posiciones desde las que se tomaron los scans 3D en el laboratorio. En la imagen, el eje X es el horizontal con el sentido positivo hacia la derecha y el Y es el vertical con el sentido positivo hacia arriba

y está circunscrito a un área muy pequeña. Por otra parte, el algoritmo de SLAM 6D trabaja con seis grados de libertad, lo que permite al mismo rotar los datos de los scans libremente respecto a cualquier eje espacial. El contar en los scans con una buena definición de las paredes, pero pobre en el suelo y el techo hace que el encaje de los múltiples scans sea bueno en su alineación respecto al plano XY, (figura 2.6) pero no tan buena respecto al YZ.

Si se observa la parte inferior de la figura 5.9), se ve una serie de «filamentos» que se desprenden del suelo. El origen de los mismos se encuentra en medidas erróneas obtenidas por el sensor láser. Estas medidas erróneas se deben, generalmente, a reflejos sobre superficies brillantes como el propio suelo o un marco de aluminio.

5.2.2. Creación de un mapa único

Tras lograr con éxito crear dos mapas para cada uno de los dos escenarios seleccionados se procedió a combinar en un único mapa los scans adquiridos en el pasillo y en el laboratorio. Para ello se han ordenado secuencialmente los scans, situando en primer lugar los scans del pasillo (scans 0 a 23) y, posteriormente, los scans del laboratorio (scans 24 a 31). También ha habido que modificar la estimación odométrica de los scans del laboratorio para aproximarlos al sistema de referencia del pasillo. Esto se ha conseguido introduciendo, a mano, una traslación en el plano XY de todas

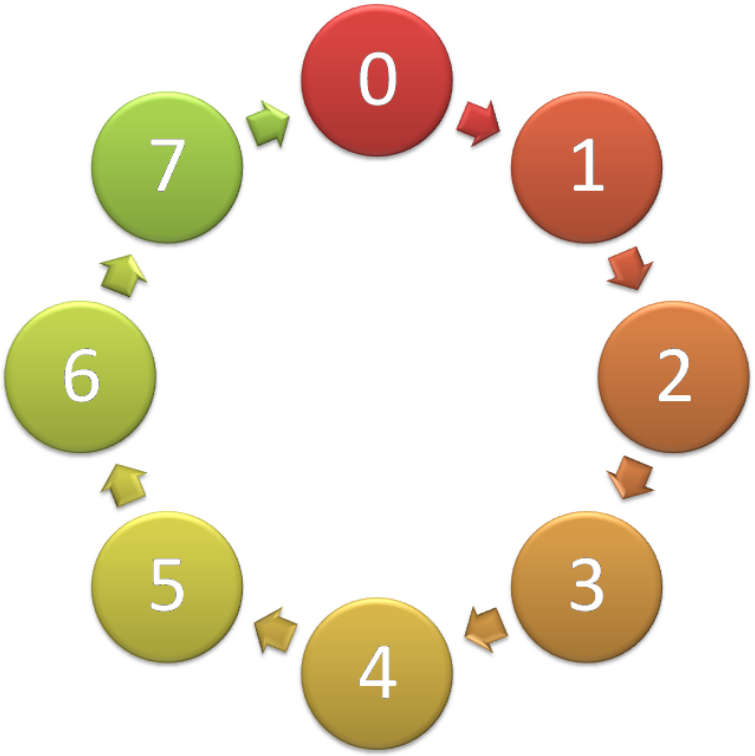

Figura 5.8: Grafo de restricciones entre las posiciones desde las que se tomaron los scans del pasillo

Figura 5.9: Proyección del pasillo sobre el plano YZ. Se observa una caída del nivel del suelo y el techo a medida que nos desplazamos hacia la derecha.

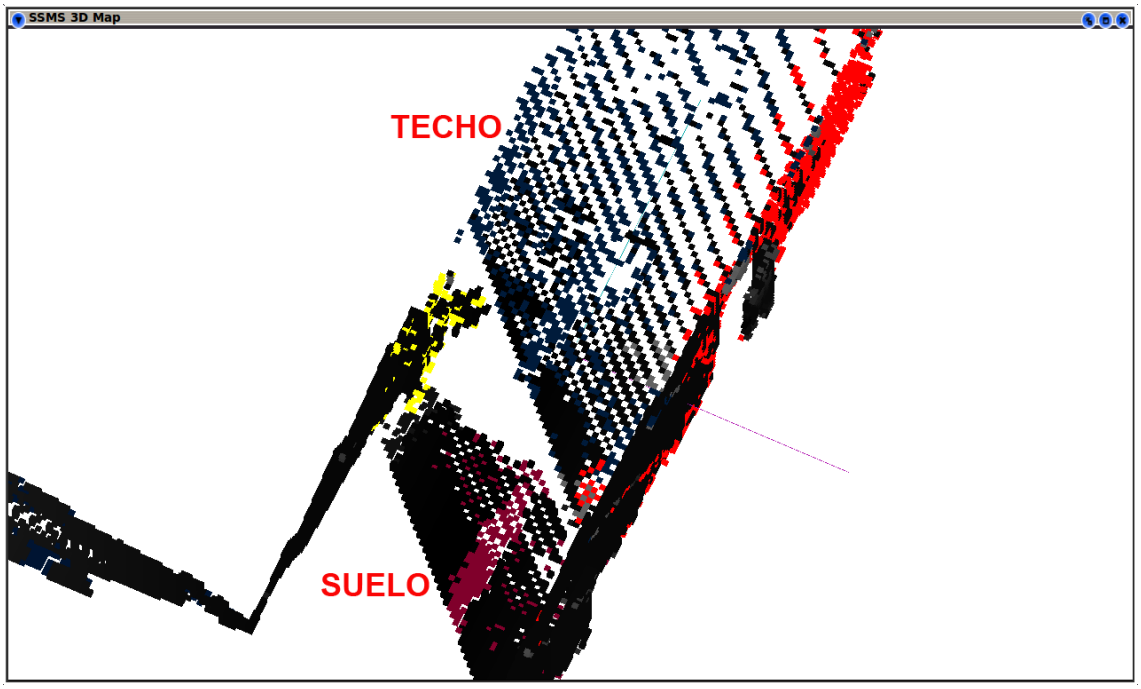

Figura 5.10: Vista del scan número 14 del pasillo. Se aprecia una pobre definición en el techo y en el suelo.

las estimaciones odométricas de los scans del laboratorio.

Se ha procedido, finalmente, a combinar ambos escenarios en un único MSM. Para ello, se ha aplicado el SLAM 6D de forma conjunta a los 32 scans, los tomados desde el pasillo y los tomados desde el laboratorio. El resultado es que el se puede ver en la figura 5.11. Las posiciones desde lasque se tomaron los scans se puede ver en la figura 5.12 y el grafo construido para llevar a cabo el SLAM 6D en la figura 5.13.

5.3. Etiquetado de estructuras del entorno

Tras el SLAM 6D se registraron en sendos MSM los scans correspondientes al pasillo y al laboratorio, como se mostró en la sección anterior. Tras crear los respectivos mapas se procedió a localizar planos mediante la aplicación del algoritmo «eficient RANSAC para Mapas de Superficie Multinivel» [11]. Los mapas localizados son marcados con distintos colores en las figuras 5.14 y 5.15 que muestran una representación tridimensional de los respectivos mapas. Por simplicidad, para mejorar la visión de los mapas, se ha eliminado de las vistas el suelo y el techo. Además de este mapa 3D, el sistema vuelca en un fichero de salida la información relativa a los planos localizados. La información proporcionada en el fichero se muestra en las tablas 5.1 y 5.2. Viendo los planos detectados en el caso del pasillo podemos observar como el único etiquetado como mesa, corresponde, por su altura, en realidad con una porción del suelo. Como se comentó anteriormente, la inclinación en el encaje de los scans 3D provoca un «desnivel» en el nivel del suelo. El algoritmo

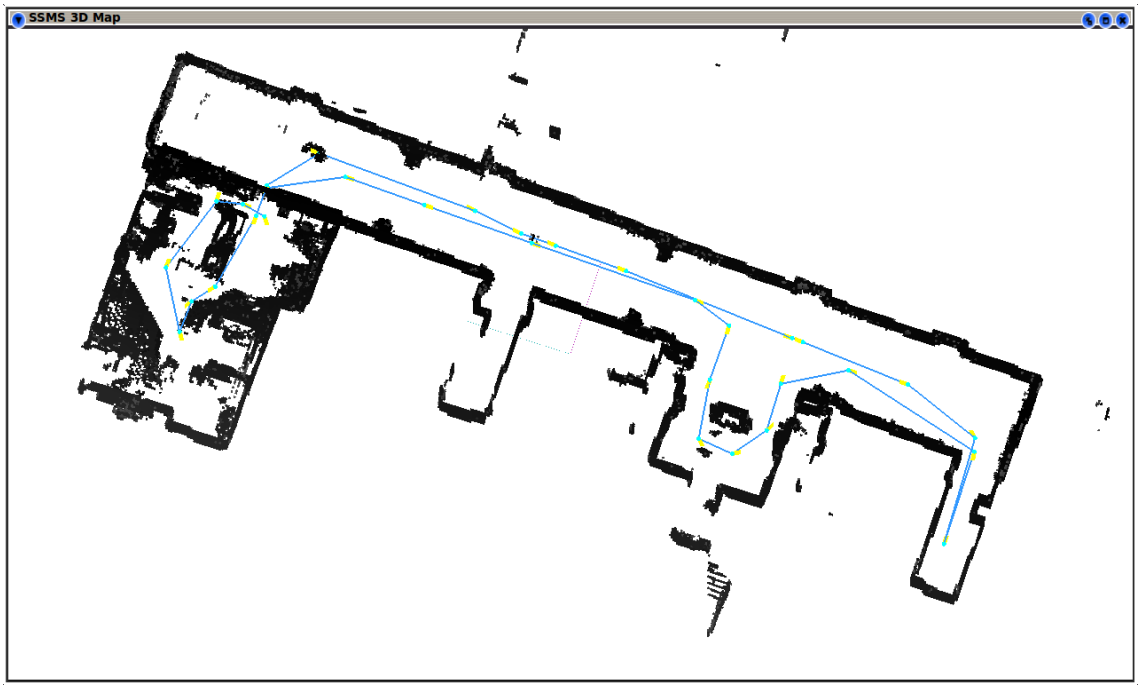

Figura 5.11: Vista aérea oblicua del mapa de superficies multinivel conjunto para el pasillo y el laboratorio tras ser corregidas las posiciones por el algoritmo SLAM 6D. La línea marca la trayectoria seguida por el robot y los puntos la posición desde la que se tomó el scan

etiqueta como suelo sólo el plano horizontal más bajo. En el caso del laboratorio está etiquetado como mesa un plano horizontal situado a 2152mm. del suelo. Este plano se corresponde con la parte inferior de una gran viga presente en el laboratorio que, a pesar de su altitud, no es el plano horizontal más alto y, por ello, está etiquetado como «mesa». En ambos casos, aunque se ha respetado la salida tal como se obtuvo, se podrían filtrar fácilmente estas falsas «mesas» añadiendo a la descripción de mesa que se debe encontrar en un determinado rango de alturas.

Una tarea no simple incluida en el listado de trabajos futuros es la determinación del contorno límite del espacio. Esta tarea, encontrar los límites del espacio en el interior de un mapa 3D, se ve dificultada por la incompletitud de la información disponible a través de los scans y del MSM. Solamente como ilustración, mostramos dos imágenes correspondientes a la envolvente conexa 2D de la proyección de los planos «pared» sobre el plano XY. En el caso del pasillo, al contar este con varios recodos y habitaciones anexas, la envolvente conexa no se aproxima al contorno real del espacio (ver figura 5.16). En el caso del laboratorio, al ser este una simple habitación, se consigue una mejor aproximación, aun así, no completamente exacta (ver figura 5.17).

Finalmente, la localización de posibles puertas. El software que hemos desarrollado muestra la localización de posibles puertas sobre un mapa 2D, correspondiente a la planta del mapa que se trate. En este mapa 2D así generado, se muestran como puntos rojos las posibles localizaciones de puertas. Vemos en la figura 5.18 el resultado de la localización de puertas en el pasillo. En el caso

Tabla 5.1: Planos detectados y etiquetados en el pasillo

Punto del plano	Normal al plano	Etiqueta
(1533.333333, -9566.666667, 2689.362731)	(-0.925947, 0.377654, 0.000000)	PARED
(3800.000000, 566.666667, 930.696148)	(-0.946334, 0.323191, 0.000000)	PARED
(-33.333333, -22300.000000, -100.000000)	(-0.002589, 0.003436, 0.999991)	MESA
(1933.333333, -34733.333333, -246.575618)	(-0.996882, 0.078909, 0.000000)	PARED
(-900.000000, -8500.000000, 2398.674538)	(0.998800, -0.048971, 0.000000)	PARED
(2700.000000, -8266.666667, -266.666667)	(-0.043107, 0.020537, -0.998859)	SUELO
(1700.000000, -8500.000000, 2700.749777)	(-0.044836, 0.027567, -0.998614)	TECHO
(1300.000000, 5333.333333, 811.455989)	(-0.043059, -0.999073, 0.000000)	PARED
(3000.000000, -16966.666667, 1800.000000)	(-0.997742, 0.051334, 0.043310)	PARED
(-800.000000, -34933.333333, -1166.666667)	(0.042873, 0.998723, -0.026715)	PARED
(-4433.333333, -25333.333333, 1360.267869)	(0.038032, 0.999277, 0.000000)	PARED
(-1433.333333, -35033.333333, 1762.934206)	(0.057462, 0.998348, 0.000000)	PARED
(3166.666667, -11700.000000, 1315.742269)	(-0.998408, 0.056401, 0.000000)	PARED
(-7400.000000, -11100.000000, 309.309588)	(0.999340, 0.036335, 0.000000)	PARED
(-2000.000000, -19733.333333, 847.281155)	(-0.978642, 0.205573, 0.000000)	PARED
(1933.333333, -11333.333333, -213.634329)	(-0.915280, 0.402819, 0.000000)	PARED
(-9400.000000, -22533.333333, 1175.285695)	(0.450780, 0.892635, 0.000000)	PARED
(-3733.333333, -12666.666667, 2158.254258)	(0.088668, 0.996061, 0.000000)	PARED
(-6366.666667, -20266.666667, 2377.537241)	(0.016571, 0.999863, 0.000000)	PARED
(-3500.000000, -26800.000000, -27.118543)	(-0.179621, 0.983736, 0.000000)	PARED

Tabla 5.2: Planos detectados y etiquetados en el laboratorio

Punto del plano	Normal al plano	Etiqueta
(6650.000000, 1733.333333, 285.079274)	(0.004850, -0.999988, 0.000000)	PARED
(-1050.000000, -1333.333333, 2158.100258)	(0.999887, -0.015044, 0.000000)	PARED
(-166.666667, 2000.000000, 102.965408)	(-0.016484, 0.012047, -0.999792)	MESA
(9233.333333, -2516.666667, 2766.666667)	(-0.014938, -0.000899, -0.999888)	TECHO
(2283.333333, -5533.333333, 1725.587462)	(0.025734, 0.999669, 0.000000)	PARED
(266.666667, -1683.333333, 450.000000)	(0.132041, 0.991231, -0.005191)	PARED
(10483.333333, 700.000000, 2350.000000)	(-0.999804, -0.013186, -0.014755)	PARED
(7433.333333, 1600.000000, 2316.666667)	(-0.037407, -0.998433, -0.041612)	PARED
(1983.333333, -4100.000000, 79.553350)	(0.133849, 0.991002, 0.000000)	PARED
(2366.666667, 2933.333333, 211.699453)	(-0.221088, -0.975254, 0.000000)	PARED
(-866.666667, 3000.000000, 2500.000000)	(0.999796, -0.019785, -0.004141)	PARED
(9866.666667, -1183.333333, 733.333333)	(-0.994441, -0.105273, -0.002339)	PARED
(4766.666667, 383.333333, 747.202182)	(-0.054243, -0.108954, -0.992566)	MESA
(-933.333333, -1966.666667, 2824.386214)	(0.998650, 0.051942, 0.000000)	PARED
(750.000000, -3283.333333, 547.152895)	(-0.747922, 0.663787, 0.000000)	PARED
(7150.000000, -3750.000000, 83.333333)	(0.231176, 0.861128, 0.452787)	MESA
(9850.000000, -4500.000000, 650.293080)	(-0.999750, 0.022339, 0.000000)	PARED
(9800.000000, -1800.000000, 1450.000000)	(-0.998873, -0.038496, 0.027764)	PARED
(1683.333333, -5283.333333, 2152.564640)	(-0.017263, 0.003966, -0.999843)	MESA
(-716.666667, -5500.000000, 2600.000000)	(0.025285, 0.999508, 0.018580)	PARED
(10600.000000, 633.333333, 2374.954747)	(-0.995033, 0.099541, 0.000000)	PARED
(2333.333333, -3366.666667, -41.717758)	(-0.098072, -0.056388, 0.993581)	SUELO
(350.000000, 2983.333333, 2716.666667)	(0.016982, -0.999856, 0.000000)	PARED

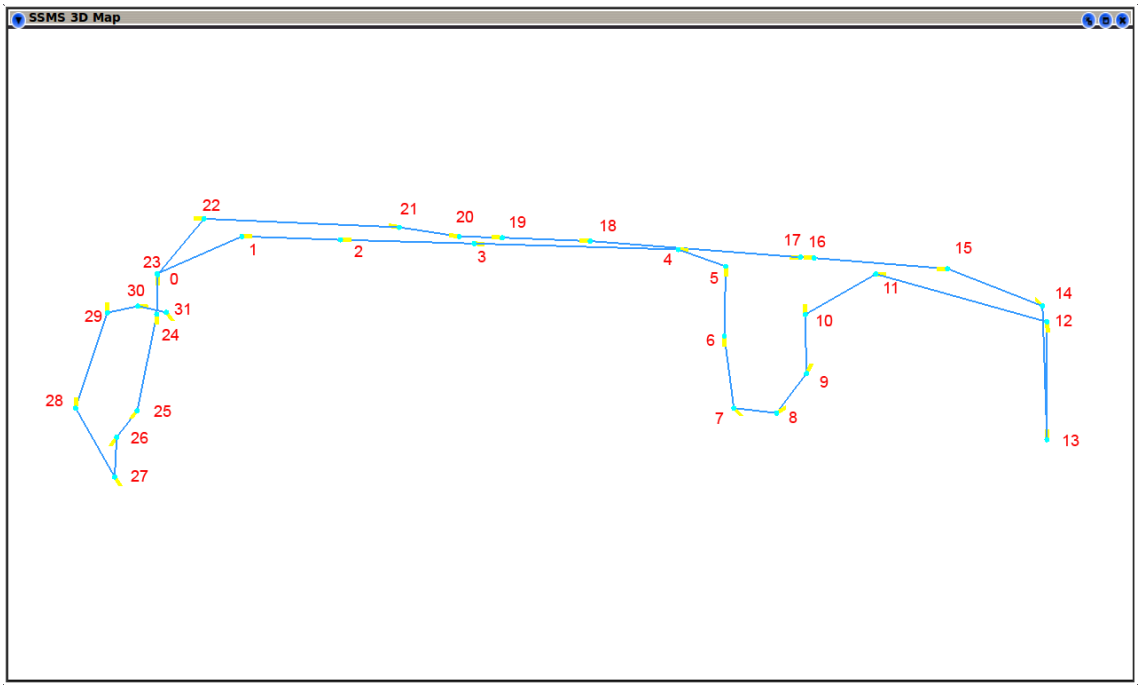

Figura 5.12: Localización de las distintas posiciones desde las que se tomaron los scans 3D en el escenario conjunto del pasillo más el laboratorio. En la imagen, el eje X es el vertical con el sentido positivo hacia arriba y el Y es el horizontal con el sentido positivo hacia la izquierda

del laboratorio, correctamente no se obtuvo ninguna detección al encontrarse la única puerta del espacio cerrada durante la adquisición de los scans.

El algoritmo presenta, como vemos en la imagen, falsos positivos debido a que el plano que forma la pared no esté completamente definido por oclusiones de otros objetos y por no haber sido totalmente barrido por el láser. En cualquier caso, como se comentó en el capítulo anterior, el interés de la detección de las posibles puertas consiste en señalar al sistema posible zonas de interés y atraer su atención para una exploración más detallada.

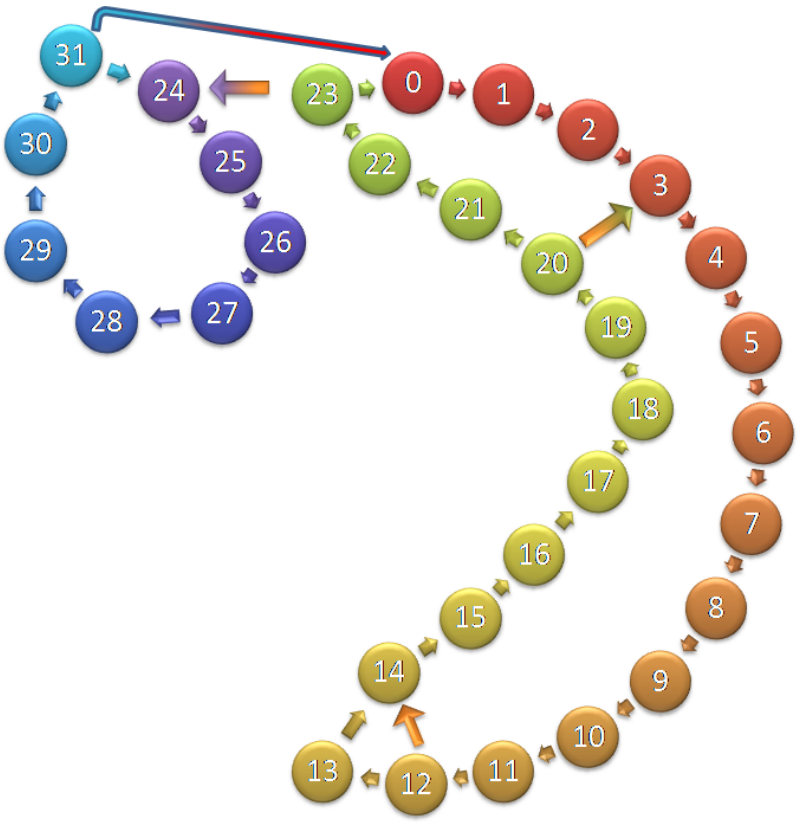

Figura 5.13: Grafo de restricciones entre las posiciones desde las que se tomaron los scans conjuntos del pasillo y el laboratorio

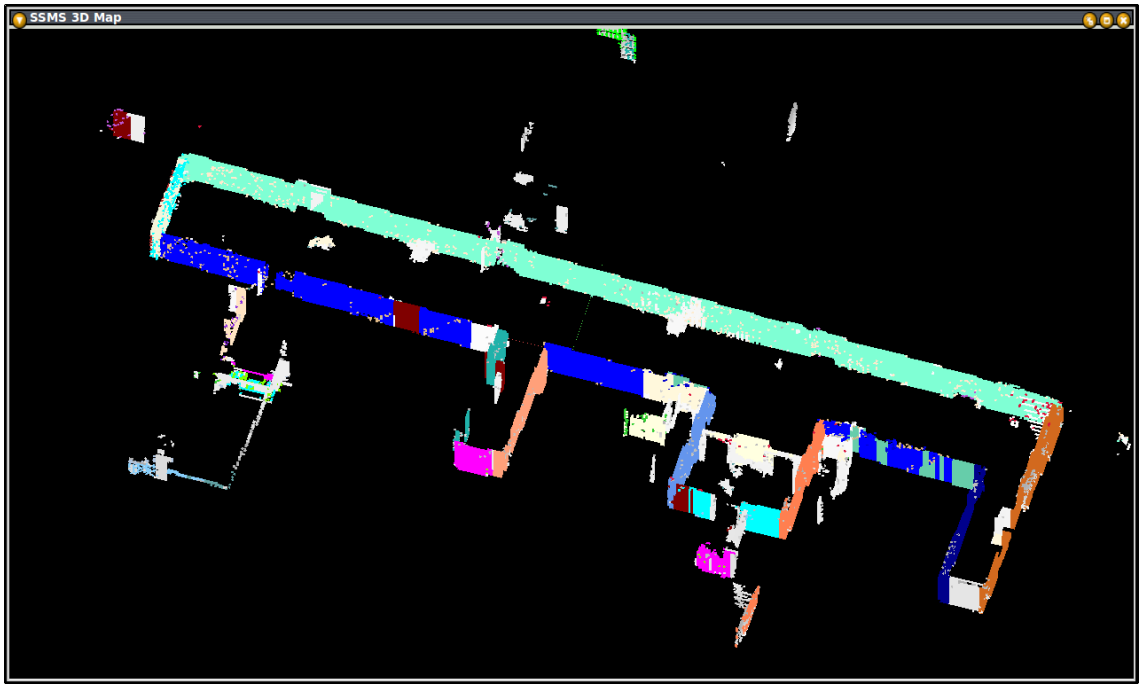

Figura 5.14: Planos localizados en el mapa del pasillo. Las zonas blancas son áreas que no han sido asignadas a ningún plano. En el resto, cada color representa un plano diferente

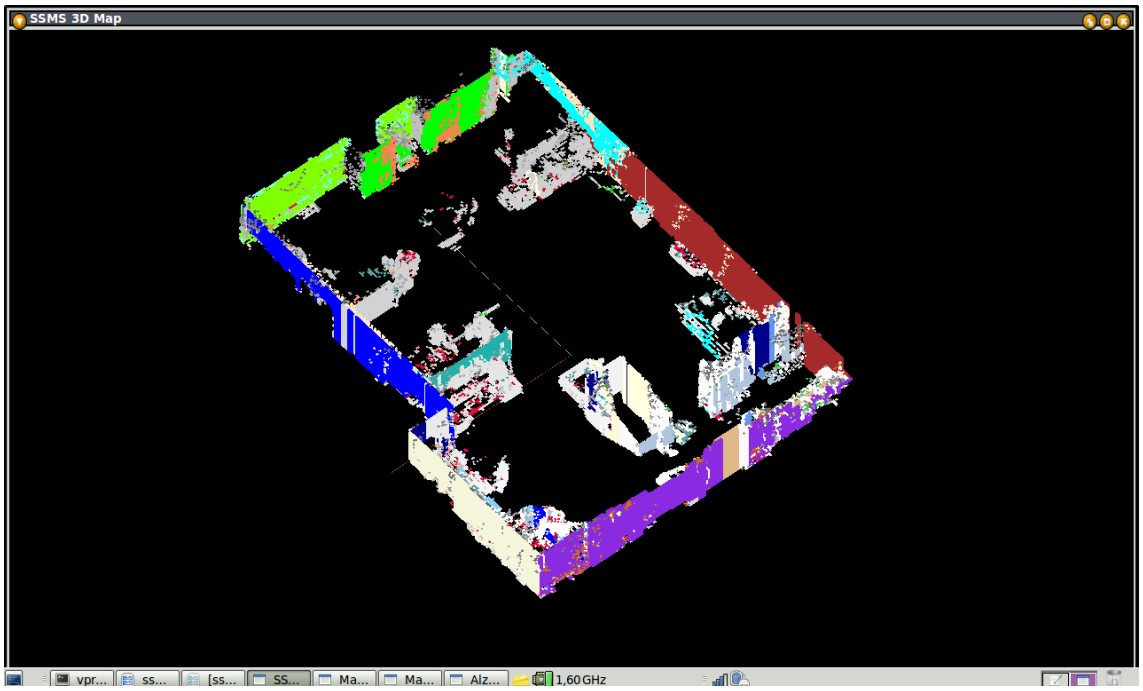

Figura 5.15: Planos localizados en el mapa del laboratorio. Las zonas blancas son áreas que no han sido asignadas a ningún plano. En el resto, cada color representa un plano diferente

Figura 5.16: Envolvente conexa (línea roja) 2D de las proyecciones de los planos «pared» sobre el plano XY en el caso del pasillo

Figura 5.17: Envolvente conexa (línea roja) 2D de las proyecciones de los planos «pared» sobre el plano XY en el caso del laboratorio

Figura 5.18: Puertas localizadas en el mapa del pasillo

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Se ha desarrollado un sistema hardware-software de adquisición de datos de rango en 3D, compuesto de un robot móvil equipado con sensor de rango por barrido láser instalado en un sistema apuntador de dos grados de libertad. Mediante un joystick inalámbrico se controlan tanto los movimientos del robot, como los del sistema apuntador y se determinan en qué momento se adquiere un barrido o scan tridimensional completo.

Los resultados alcanzados en el PFC [11] se han generalizado a un entorno de interior realista donde el sistema sensor se desplaza. La construcción de mapas a partir de múltiples vistas se ha resuelto mediante la librería SLAM 6D. El procedimiento de construcción del mapa se ha resuelto en dos fases: una local, basada en el algoritmo ICP, donde se miminiza el error de encaje de dos barridos próximos; y otra global donde, a partir de la red de poses desde la que se han obtenido barridos, se corrige globalmente el mapa siguiendo un criterio de máxima verosimilitud. Una parte significativa del trabajo experimental desarrollado durante este proyecto se ha tenido que dedicar a estudiar, conocer y ajustar esta herramienta de SLAM ya que la documentación es escasa y fragmentaria.

Se ha demostrado cómo es posible construir un mapa de forma incremental a partir de mapas de espacios contiguos obtenidos de forma independiente. De esta forma, es posible mapear un espacio complejo, compuesto de múltiples estancias, a partir de los mapas obtenidos en cada una de las estancias. La segmentación de un espacio complejo en estancias establece un mecanismo simple y eficaz de gestionar la navegación en dicho espacio al requerir de mapas de menores dimensiones. Se ha demostrado que no es necesario describir los mapas parciales con la misma resolución para que éstos puedan formar un mapa del espacio global.

El mapa de superficies multinivel que se obtiene permite una descripción compacta de entornos tridimensionales de grandes dimensiones, como se ha demostrado en este proyecto. Además, la propia naturaleza discreta de los mapas de superficie multinivel ha demostrado ser un índice potente para ponderar la calidad o convergencia del procedimiento SLAM de construcción del mapa global.

Empleando el algoritmo eRANSAC sobre la superficie multinivel ha sido posible identificar planos y etiquetar éstos como suelo, techo, pared, mesa o puerta. De esta forma se han dado los primeros pasos para construir una descripción simbólica del entorno de alto nivel semántico.

6.2. Trabajo futuro

A partir del trabajo expuesto aquí y de la experiencia adquirida se abren distintas vías de trabajo futuro:

- Determinación de los contornos y límites del espacio. Sería de utilidad para la navegación que el sistema pudiera «saber» cuáles son los límites del espacio que lo circunscribe y cuáles son las zonas de tránsito entre los distintos segmentos (habitaciones) del espacio.
- Construcción de una base de datos con descripción de objetos. En el presenta trabajo, la detección de estructuras se ha hecho implementando algoritmos ad hoc. El diseño de un sistema de descripción de estructuras, junto con un motor de búsqueda que operara sobre el MSM permitiría incrementar la versatilidad de la herramienta.
- Asignación de texturas. El mapa de superficies multinivel que se construye podría enriquecerse incorporando textura a las estructuras del mismo. Esta textura se obtendría de una cámara incorporada al robot. La textura serviría para mejorar la detección de objetos y la representación del mapa.
- Planificación de trayectorias 3D. En este trabajo se ha desplazado al robot únicamente a lo largo una superficie horizontal. Esto ha estado condicionado por los sensores odométricos del robot usado, los cuáles solo proporcionan la traslación en el plano XY y la orientación angular respecto al eje Z. Dotar al robot de un juego mayor de sensores, inclinómetros, sensores de inercia, etc, haría posible al sistema el transitar por distintos niveles unidos por rampas, ascensores, etc y así construir un mapa más completo. Una vez construido un mapa coherente 3D, se podría usar para planificar trayectorias que impliquen distintas alturas como el paso de rampas, etc.
- Ejecución de SLAM en tiempo real. Un trabajo de futuro sería la realización de la corrección odométrica «on-line» mientras el robot se desplaza.

Bibliografía

[1] Anthony Stentz, Dieter Fox, Michael Montemerlo, and Michael Montemerlo. Fastslam: A factored solution to the simultaneous localization and mapping problem with unknown data association. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598. AAAI, 2003.

[2] Shu-Ping Zhang, Yong-Sheng Ding, and Kuang-Rong Hao. An immune evolutionary algorithm based pose estimation method for parallel manipulator. In *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 745–750, New York, NY, USA, 2009. ACM.

[3] Dario Lodi Rizzini and Stefano Caselli. Metric-topological maps from laser scans adjusted with incremental tree network optimizer. *Robot. Auton. Syst.*, 57(10):1036–1041, 2009.

[4] Andreas Nüchter. *3D Robotic Mapping*. Springer, 2009.

[5] P. Pfaff; R. Triebel; W. Burgard. An efficient extension to elevation maps for outdoor terrain mapping and loop closing. *International Journal of Robotics Research*, 2007.

[6] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[8] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[9] Klein R. Schnabel R., Wahl R. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26:214–226, 2007.

[10] Sebastian Thrun. Simultaneous localization and mapping. In Margaret E. Jefferies and Wai-Kiang Yeap, editors, *Robotics and Cognitive Approaches to Spatial Mapping*, volume 38 of *Springer Tracts in Advanced Robotics*, pages 13–41. Springer, 2008.

[11] Víctor Prieto Marañón. Proyecto final de carrera: Construcción de un mapa tridimensional a partir de un sensor láser activo. Master's thesis, Facultad de Informática. Universidad de Las Palmas de Gran Canaria, 2009.

[12] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans ASME J. Appl. Mech*, 23:215–221, 1955.

[13] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4(4):333–349, 1997.

[14] Dorit Borrmann, Jan Elseberg, Kai Lingemann, Andreas Nüchter, and Joachim Hertzberg. Globally consistent 3d mapping with scan matching. *Robot. Auton. Syst.*, 56(2):130–142, 2008.

[15] P.J. Besl and N.D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[16] Kai Lingemann Joachim Hertzberg Jochen Sprickerhof, Andreas Nüchter. An explicit loop closing technique for 6d slam. In *n Proceedings of the 4th European Conference on Mobile Robots (ECMR '09)*, 2009.

[17] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *Int. J. Rob. Res.*, 25(5-6):403–429, 2006.

[18] Joachim Hertzberg Andreas Nüchter, Kai Lingemann. *6D SLAM Simultaneous 6 D.O.F. Localization and 3D Mapping*. Department of Mathematicas/Computer Science. Institude of Computer Science Knowledge-Base Systems Research Group. University of Osnabrück, June 2009.

[19] Andreas Nüchter. *SLAM Tutorial 6D SLAM*. Univerity of Osnabrück, 2007.