DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Estimación de la covarianza de ICP aplicada a la localización de robots móviles

Noviembre 2016

Andrés Stepaniuk
andres.stepaniuk@gmail.com

Directores

Pablo De Cristóforis
pdecris@gmail.com

Thomas Fischer
tfischer@dc.uba.ar

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Estimación de la covarianza de ICP aplicada a la localización de robots móviles

En robótica móvil, el problema de la localización consiste en la estimación de la ubicación del robot respecto de un marco de referencia. En los métodos de localización absoluta, se calcula utilizando un sistema de referencia o mapa conocido de antemano, y sensores que permiten estimar instantáneamente la ubicación del robot respecto de ese sistema de referencia global. Por otro lado, los métodos de localización relativa (también conocida como *dead-reckoning*) se basan en la idea de ir estimando el desplazamiento del robot en forma incremental, a partir de la posición estimada en un instante anterior.

El método más clásico de localización relativa es la odometría, que consiste en el uso de encoders asociados a los motores del robot. Para mejorar esta estimación se pueden utilizar sensores exteroceptivos que midan la distancia del robot a los objetos en el ambiente. Los telémetros láser permiten obtener una nube de puntos que representan las distancias a los objetos. Con esta información es posible calcular el desplazamiento del robot a partir de estimar la transformación que lleva la nube de puntos de un sensado a la nube de puntos correspondiente al sensado en el instante siguiente. Para calcular la transformación entre dos nubes de puntos existen diversos algoritmos, dentro de los cuales puede destacarse ICP (*Iterative Closest Point*) por ser uno de los más utilizados.

Para poder fusionar la información de odometría con la información del sensor láser podemos usar el Filtro Extendido de Kalman. Para utilizar este método de fusión es necesario estimar la covarianza de cada fuente de información. Sin embargo, el algoritmo ICP no devuelve la covarianza asociada a su estimación.

En esta tesis se presenta un método para estimar la covarianza del algoritmo ICP para poder fusionar de manera más ajustada la información obtenida con el sensor láser con la información provista por odometría utilizando el Filtro Extendido de Kalman. Los resultados obtenidos muestran que este método de fusión de sensores resulta en una estimación más precisa de la pose del robot en comparación con métodos que mantienen fija la covarianza del algoritmo ICP.

ICP covariance estimation applied to mobile robot localization

In mobile robots, the problem of robot localization consists in estimating the position of the robot according to a frame of reference. In absolute localization methods, the position is calculated using a known map or reference, and sensors that enable instantaneous estimation of the position of the robot according to a global reference. On the other hand, relative localization methods (also known as dead-reckoning) are based on the idea of estimating the movement of the robot incrementally, starting from the estimated position in a previous moment in time.

One of the best known methods of relative localization is wheel odometry, which consists in using encoders attached to the motors of the robot . Exteroceptive sensors can be used to improve this estimation, which measure the distance from the robot to different objects in the environment. Laser telemetry allows the robot to obtain a point cloud representing distances to objects. Using this information, it's possible to calculate the movement of the robot by estimating the transformation that maps one point cloud to another point cloud forward in time. Various methods exist to compute the transformation between two point clouds, among them we can mention Iterative Closest Point, or ICP, for being one of the most widely used.

The Extended Kalman Filter can be used to combine the information from wheel odometry with the information coming from the laser scanner. To be able to use this method we need to estimate the covariance of each information source. However, ICP does not provide the covariance associated to its estimation of movement.

In this thesis we present a method to estimate the covariance for the ICP algorithm, to be able to use EKF to precisely fuse the information provided by the laser sensor and the information provided by wheel odometry. Experiments show that this method of sensor fusion results in a more precise estimation for the robot's pose, compared to methods using a fixed covariance for ICP.

Índice general

Capítulo 1

Introducción

1.1 Motivación

En aplicaciones de robótica móvil resulta indispensable abordar el problema de la localización [1]. Si un robot no puede saber con seguridad en qué lugar se encuentra, difícilmente pueda planificar y ejecutar correctamente la siguiente acción [2]. Cuando hablamos de ubicación nos referimos tanto a la posición como a la orientación del robot respecto de un sistema de coordenadas.

El problema de la localización puede ser abordado en forma absoluta o relativa. En los métodos absolutos, se localiza al robot respecto de un sistema de referencia o mapa, utilizando sensores exteroceptivos que permiten medir u observar directamente el entorno en cada instante de tiempo. Estas observaciones del ambiente proveen información acerca de la ubicación del robot, independientemente de cualquier estimación de una ubicación anterior. No obstante, estos métodos dependen de información previa e infraestructura específica (como es el caso del Sistema de Posicionamiento Global o GPS), y no suelen ser muy precisos en la navegación a corto plazo. Los métodos de localización relativa (también conocidos como de *dead-reckoning*) se basan en la idea de estimar la posición actual en forma incremental, a partir de la integración del desplazamiento respecto de la posición anterior. Un ejemplo clásico de *dead-reckoning* consiste en utilizar *encoders,* que permiten traducir el giro de las ruedas durante un intervalo de tiempo en un desplazamiento lineal y angular del robot, técnica conocida como *odometría*. Sin embargo, por su naturaleza integrativa, los métodos de *dead-reckoning* son propensos a acumular error en forma no acotada.

La odometría basada en encoders es una de las formas más comunes y simples de localización relativa. La premisa de esta forma de localización es que el movimiento de las ruedas es directamente traducible al desplazamiento del robot. Sin embargo, en la práctica esto no se sostiene y aparecen errores de estimación, que pueden dividirse en errores sistemáticos y no sistemáticos [3]. Los errores sistemáticos son producto de imperfecciones constructivas en el vehículo, tales como ruedas desiguales o errores en la estimación de las medidas involucradas en las ecuaciones de odometría (radio de las ruedas, distancia entre ruedas, etc.). Los no sistemáticos, en cambio, resultan de la interacción con el ambiente, como puede ser cuando las ruedas patinan sobre una superficie sin que el robot realmente avance o cuando existen irregularidades en el terreno que invalidan el modelo de movimiento considerado (que asume un terreno plano). Si bien existen técnicas efectivas para caracterizar y corregir los errores sistemáticos [4], los no sistemáticos no pueden ser corregidos mediante una calibración previa ya que estos dependen del entorno. Por esta razón, se suelen incorporar sensores exteroceptivos para detectarlos y compensarlos.

Entre los primeros sensores exteroceptivos que se emplearon para abordar el problema de la localización a partir de mediciones de distancia a los objetos se pueden mencionar los sonares de ultrasonido y los telémetros de luz infrarroja [5]. Sin embargo, estos sensores son muy imprecisos

y propensos al ruido. Posteriormente, surgieron los sensores basados en tecnología láser, los cuales son mucho más precisos que los anteriores [6, 7]. En los últimos años, se ha visto un incremento de la utilización de cámaras como sensores especializados para resolver los problemas de la localización, navegación y construcción de mapas del entorno[8, 9]. Bajo este enfoque, las imágenes obtenidas por una cámara montada en el robot son procesadas y comparadas en cada instante con las imágenes anteriores, lo que permite deducir el desplazamiento del robot. Sin embargo, es más costoso computacionalmente obtener mediante una cámara información equivalente a la de un sensor de rango, ya que requiere del procesamiento de las imágenes capturadas. Por este motivo, el uso de lásers para sistemas de localización relativa sigue siendo una buena opción.

Los telémetros láser (o sensores láser) son capaces de medir distancias a los objetos del entorno con una resolución lineal de pocos milímetros. Ante cada sensado, se obtiene una nube de puntos que representa los elementos detectables del entorno (dentro del rango de observación del sensor, para el caso de un láser 2D). A partir de estimar la transformación rígida que transforma una nube en la otra, es posible estimar el desplazamiento del robot durante ese intervalo de tiempo. Este proceso es conocido como registración del muestreo (o escaneo) del láser, o *scan matching* [10]. Para abordar este problema, existen diversos algoritmos entre los cuales el más utilizado es *Iterative Closest Point* [11, 12](o ICP).

Si bien ICP puede dar muy buenos resultados para resolver el problema de la registración por láser, este algoritmo asume que siempre habrá algún cambio perceptible de los objetos u elementos del entorno. En ciertos contextos esta hipótesis no es cierta (por ejemplo un robot moviéndose en forma recta por un pasillo recto y uniforme) por lo que no alcanza con usar un láser con ICP para obtener un sistema fiable de localización relativa. Para resolver este problema, es posible fusionar la estimación obtenida con el láser y el algoritmo ICP, con la obtenida mediante otros sensores (por ejemplo, con la odometría basada en encoders). Esto se pueda hacer utilizando algún tipo de filtro probabilístico, entre los cuales se destaca el Filtro Extendido de Kalman (EKF) [13, 14]que es ampliamente utilizado en este tipo de problemas.

El Filtro Extendido de Kalman es una versión no lineal del Filtro de Kalman. El mismo integra sucesivas observaciones a través del tiempo y provee una predicción probabilística del estado en todo momento. La estimación de estado se basa en conocer, además de las observaciones, el modelo que estamos usando para representar el estado, junto con las incertezas propias de éste y de los errores en las observaciones proporcionadas por los sensores.

Para utilizar efectivamente el Filtro Extendido de Kalman para fusionar la información de ubicación, es necesario contar con un modelo preciso de las incertidumbres provenientes de cada fuente de información. Éstas se ven reflejadas en la incertidumbre de cada predicción del estado, representada a partir de una matriz de covarianza.

1.2 Objetivos

El objetivo de esta tesis es desarrollar un método para el cálculo de la incertidumbre asociada a las estimaciones obtenidas por el algoritmo de Iterative Closest Point (ICP). Por otro lado, también queremos modelar la incertidumbre producida por la odometría basada en encoders.

Utilizando estos modelos probabilísticos, se desea integrar la información provista por ambos sensores en un EKF para obtener una estimación más robusta de la pose del robot. De ésta manera se espera mejorar la localización para robots que cuentan con encoders asociados a las ruedas como sensores propioceptivos y telémetro láser como sensor exteroceptivo.

1.3 Organización de la tesis

Este trabajo está organizado de la siguiente manera. En Preliminares (Cap. 2) se presenta el marco teórico necesario para comprender el trabajo desarrollado, convenciones sintácticas y de representación, y los conceptos básicos sobre los cuáles se basa el desarrollo de esta tesis como son: la odometría basada en encoders, la registración por ICP de sensado láser y la fusión de sensores utilizando el Filtro Extendido de Kalman. Luego, en Método (Cap. 3) se describe en detalle el método desarrollado para lograr el objetivo en cuestión: contar con un modelo de las incertidumbres provenientes de la registración de láser por ICP, para lo cual necesitamos estimar la matriz de covarianza del resultado del algoritmo. En Implementación (Cap. 4) se detalla la implementación del método propuesto utilizando ROS (*Robot Operating System*), el paquete *robot_ localization* y la librería *libpointmatcher,* así como el diseño de los nodos implementados y la comunicación entre estos. En Experimentación y Resultados (Cap. 5) se describen los materiales y métodos empleados para realizar los experimentos, se presentan los resultados obtenidos tanto para la localización del robot como para la construcción de mapas del entorno a partir de conocer la pose del robot y se analiza la precisión del método de localización propuesto. Finalmente, en Conclusiones (Cap. 6) se comenta en retrospectiva cuál es la relevancia de los resultados obtenidos y se plantean potenciales trabajos relacionados a desarrollar en un futuro.

Capítulo 2

Preliminares

2.1 Odometría basada en encoders

La odometría basada en encoders es una de las técnicas más utilizadas en sistemas de localización relativos para robots sistemas de locomoción basados en ruedas. Usualmente, el movimiento de éste tipo de robots puede ser caracterizado por un modelo denominado de tracción denominado diferencial. El sistema de tracción diferencial consiste en dos ruedas con motores independientes, situadas a ambos lados del cuerpo del robot móvil. Variando la diferencia relativa de velocidad angular entre ambas ruedas el robot puede cambiar su dirección sin necesidad de un mecanismo adicional. Si ambas ruedas giran en el mismo sentido y a la misma velocidad, el robot se desplazara en linea recta, por otro lado, si las ruedas giraran a la misma velocidad pero en direcciones opuestas, el robot girara sobre el punto central entre ambas ruedas.

Aún cuando la mayoría de los robots no respetan estrictamente este modelo, por ejemplo en el caso de los que utilizan cuatro ruedas u orugas, el modelo diferencial se aplica de todas maneras por una cuestión de simplicidad.

Las ecuaciones de movimiento utilizadas en este modelo suponen entonces que los movimientos se realizan a partir del control independiente de dos motores, y que el desplazamiento de las ruedas es directamente traducible al desplazamiento real del robot, como se aprecia en la figura 2.1.

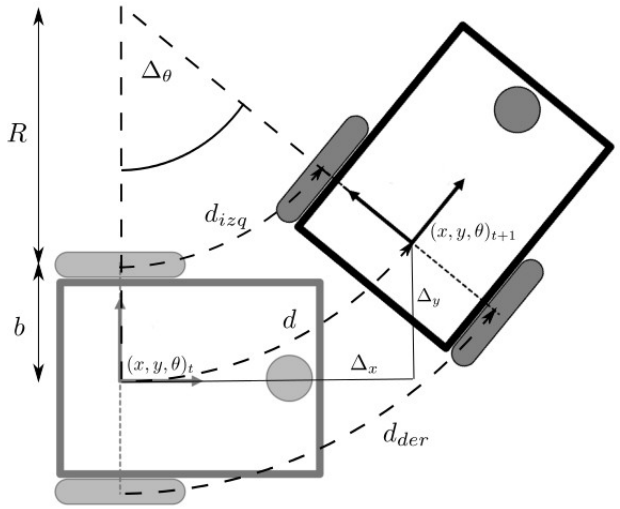


Figura 2.1: Modelo de movimiento simple para vehículo de tracción diferencial

Sin embargo, en la práctica estas suposiciones no son necesariamente ciertas. Es por esta razón, que se genera un error acumulativo y no acotado, debido a la estimación de carácter incremental de la pose.

A continuación se presenta el modelo teórico clásico de odometría para vehículos de tracción diferencial. El método estima, en cada intervalo de tiempo, el desplazamiento lineal d_{izq}, d_{der} de las ruedas, a partir de sus desplazamientos angulares α_{izq},α_{der}. A partir de estas muestras, y conociendo el diámetro D de las ruedas (que supondremos iguales), podemos calcular la distancia lineal recorrida por cada una como

$$d_{izq} = \pi D \alpha_{izq} \tag{2.1}$$

$$d_{der} = \pi D \alpha_{der} \tag{2.2}$$

A su vez, los desplazamientos angulares son calculados a partir de pulsos p_{izq} y p_{der}, generados por los *encoders* ubicados en los motores, previo a cualquier reducción mecánica, según

$$\alpha_{izq} = \frac{p_{izq}}{P} G \tag{2.3}$$

$$\alpha_{der} = \frac{p_{der}}{P} G \tag{2.4}$$

En este caso, P es la cantidad de pulsos de encoder generados por cada revolución del eje del motor, y G es el factor de reducción entre el motor y la rueda.

A partir del desplazamiento lineal realizado por cada rueda, el modelo de odometría estima el desplazamiento del robot en el último intervalo de la siguiente manera:

$$\Delta\theta = \frac{(d_{der} - d_{izq})}{2b} \tag{2.5}$$

$$\Delta x = d\cos(\theta) \tag{2.6}$$

$$\Delta y = d\sin(\theta) \tag{2.7}$$

donde b es la distancia de cada rueda al centro del robot, $\Delta\theta$, Δx, Δy son los desplazamientos angular y lineal del vehículo respectivamente y d es la distancia recorrida por el mismo.

El modelo supone que, a un intervalo de tiempo lo suficientemente chico, se puede aproximar la trayectoria del vehículo mediante un arco de radio R y ángulo $\Delta\theta$, como se aprecia en la figura 2.1. De aquí se desprenden las siguientes relaciones geométricas para la trayectoria de cada rueda:

$$d_{izq} = R\Delta\theta \tag{2.8}$$

$$d_{der} = (R + 2b)\Delta\theta \tag{2.9}$$

Para determinar $\Delta\theta$ (2.5), despejamos R en ambas ecuaciones, obteniendo la siguiente equivalencia:

$$\frac{d_{izq}}{\Delta\theta} = \frac{d_{der}}{\Delta\theta} - 2b$$

Por último, para encontrar d, calculamos la longitud del arco descrito por el movimiento del vehículo:

$$d = \Delta\theta\,(R + b)$$

donde, reemplazando por las ecuaciones (2.8) y (2.9), obtenemos

$$d = \frac{d_{izq} + d_{der}}{2} \tag{2.10}$$

Finalmente, la pose del robot $(x,\ y,\ \theta)_{k+1}$ en el $k+1$ se calcula de forma incremental a partir del intervalo anterior k:

$$\begin{aligned} \theta_{k+1} &= \theta_k + \Delta\theta_k && (2.11)\\ x_{k+1} &= x_k + d_{k+1} \cdot \cos(\theta_k + \Delta\theta_k) && (2.12)\\ y_{k+1} &= y_k + d_{k+1} \cdot \sin(\theta_k + \Delta\theta_k) && (2.13) \end{aligned}$$

Usando estas ecuaciones, podemos escribir la pose actual del robot como un vector $p_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^t$, y la pose actualizada como $p_{k+1} = f\,(x, y, \theta, d_{izq}, d_{der})$ de la siguiente manera:

$$p_{k+1} = f\,(x, y, \theta, d_{izq}, d_{der}) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{d_{izq}+d_{der}}{2} \cdot \cos\left(\theta + \frac{d_{izq}+d_{der}}{2}\right) \\ \frac{d_{izq}+d_{der}}{2} \cdot \sin\left(\theta + \frac{d_{izq}+d_{der}}{2}\right) \\ \frac{d_{izq}+d_{der}}{2} \end{bmatrix} \tag{2.14}$$

Como se dijo anteriormente, esta estimación incremental genera un error acumulativo, y sólo puede dar una estimación aproximada de la pose del robot. Debido a los errores de integración de las incertezas de la pose actual, y los errores provocados por el movimiento incremental $(d_{izq}; d_{der})$, el error de posición basado en la integración de la odometría crece ilimitadamente con el tiempo.

Se quiere entonces establecer un modelo de error para la pose p_{k+1} estimada por la odometría y obtener una matriz de covarianza $\Sigma_{p_{k+1}}$ para la pose misma. Para esto, se asume que así como la pose anterior p_k es conocida, también lo es la matriz de covarianza asociada Σ_{k_p}. Entonces, para un movimiento incremental $(d_{izq}; d_{der})$, asumimos la siguiente matriz de covarianza Σ_d:

$$\Sigma_d = covar\,(d_{izq}, d_{der}) = \begin{bmatrix} k_{der}\,|d_{der}| & 0 \\ 0 & k_{izq}\,|d_{izq}| \end{bmatrix} \tag{2.15}$$

donde d_{izq} y d_{der} son las distancias recorridas por cada rueda, y k_{izq} , k_{der} son constantes de error que representan parámetros no determinísticos de la transmisión del motor y la interacción de las ruedas con el suelo.

Como podemos ver, en la ecuación (2.15), asumimos lo siguiente:

- Los errores de las ruedas son independientes.
- La varianza de los errores de las ruedas izquierda y derecha son proporcionales al valor absoluto de las distancias recorridas.

Estas suposiciones, si bien no son perfectas, suelen ser suficientemente buenas. Los valores de las constantes k_{izq} y k_{der} dependen del robot y del entorno, y deben ser establecidas individualmente en cada caso de forma experimental.

Si suponemos que p y $d_{di} = (d_{der}; d_{izq})$ no están correlacionadas y que la derivada de f (2.14) se puede aproximar razonablemente por la función de Taylor de primer orden, utilizando la ley de propagación del error, tenemos que:

$$\Sigma_{p_{k+1}} = \nabla_p f \cdot \Sigma_{p_k} \cdot \nabla_p f^T + \nabla_{d_{di}} f \cdot \Sigma_d \cdot \nabla_{d_{di}} f^T$$

La matriz de covarianza $\Sigma_{p_{k+1}}$ es entonces una función de la matriz Σ_{p_k} obtenida en el paso anterior. Para la pose inicial, que se corresponde con el origen de coordenadas y sobre la cuál tenemos certeza absoluta, se puede elegir una matriz de covarianza inicial suficientemente pequeña.

Podemos calcular entonces los jacobianos necesarios $F_p = \nabla_p f$ y $F_{d_{di}} = \nabla_{d_{di}} f$ de la siguiente manera:

$$F_p = \nabla_p f = \nabla_p (f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -d\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & d\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix}$$

$$F_{d_{di}} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{d}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{d}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{d}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{d}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix}$$

2.2 Telemetría Láser

El sensor de telemetría láser es un dispositivo que utiliza uno o varios haces de luz para medir distancias. Este sensor consiste de un emisor que ilumina el objeto a medir con un haz de luz colimada (es decir, un láser), y un receptor capaz de detectar la luz reflejada en el objeto. La distancia se calcula midiendo el tiempo que tarda el haz de luz en llegar al objeto y volver. A la velocidad de la luz, esta recorre 1mm en apenas 3,3 picosegundos aproximadamente, de forma que se necesitaría una resolución temporal de al menos 6,6 picosegundos para obtener una resolución espacial de 1mm. Por lo tanto, se requieren circuitos electrónicos capaces de medir tiempos de vuelo con una resolución de pocos picosegundos. Para salvar estos problemas, los fabricantes de telémetros láser emplean distintas estrategias, como emitir pulsos largos o promediar varias mediciones.

En un sensor láser típico de un robot, se miden distancias desde un punto fijo hacia un numero de determinadas direcciones que abarcan normalmente un ángulo de apertura de 180°, 270° o 360°. Para ello se utiliza un espejo móvil rotativo delante del emisor y receptor del láser que realiza barridos sincronizados. Estos barridos pueden ser solo horizontales para un resultado bidimensional u horizontales y verticales para un resultado tridimensional.

La información se recibe en forma de arreglos de mediciones, cada uno correspondiente a un barrido completo del rango disponible. Esta información consiste en los ángulos y las distancias muestreadas a los objetos del ambiente para cada ángulo. La precisión de un sensor láser está principalmente determinada por el radio de alcance o rango, la resolución lineal y la resolución angular. La potencia del emisor determina la distancia máxima de sensado. Por otro lado, debido a su construcción, existe también una distancia mínima a partir de la cual es capaz de obtener mediciones válidas. Por último, es usual que el barrido de medición no abarque toda la circunferencia definida por el radio de alcance del sensor, sino que exista una porción que no puede ser observada. Por estas razones, el área de sensado efectiva puede ser definida tanto por distancia mínima y máxima de detección, como por ángulo mínimo y máximo, denominado *campo de visión* del sensor (Figura 2.2).

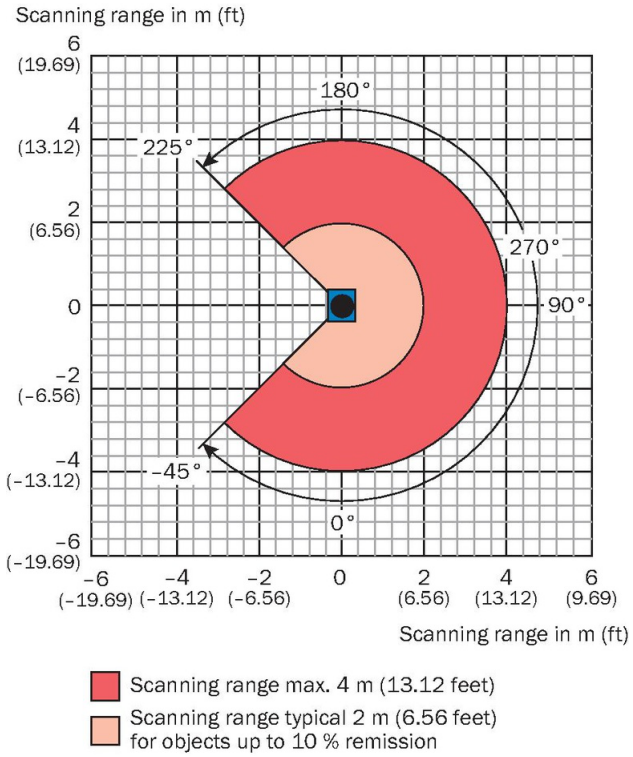


Figura 2.2: Campo de visión de un láser

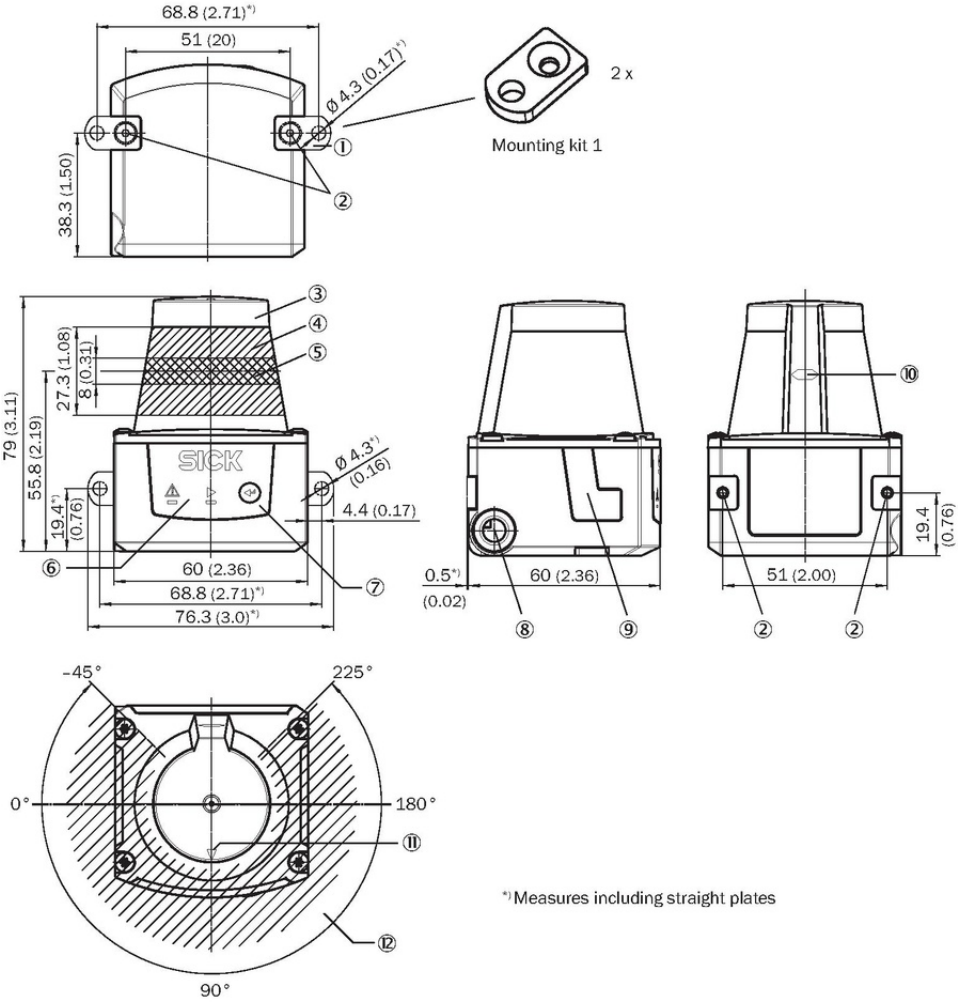


Figura 2.3: Estructura típica de un sensor láser

A partir de la información recolectada por el telémetro láser, que viene dada en coordenadas polares (distancia y ángulo de cada haz), es posible construir una nube de puntos cartesianos del entorno. Sin embargo, es usual realizar un preprocesamiento de los datos, principalmente debido al ruido de medición. El filtro más simple y efectivo, y que fue utilizado en esta tesis, es el filtro por distancia. En este caso, se define una distancia mínima y máxima que delimitan el rango aceptable para las mediciones. La distancia mínima generalmente permite evitar sensar la estructura propia del robot. Por otra parte, la distancia máxima permite fácilmente ignorar las mediciones correspondientes a elementos más distantes, que generalmente tienen asociadas un mayor ruido de medición. De todas formas, los parámetros de distancia máxima y mínima deben ser establecidos en cada caso particular.

2.3 Registración del escaneo láser

Cada escaneo es modelado mediante una nube de puntos $P \in \mathbb{R}^{3\times n}$, compuesta de n puntos de la forma:

$$\mathbf{x}_i = (x_i,\, y_i\,, 1),$$

con lo cual P queda definida de la siguiente forma:

$$\mathrm{P} = \begin{pmatrix} \mathrm{x}_1 & \mathrm{x}_2 & \cdots & \mathrm{x}_\mathrm{n} \\ \mathrm{y}_1 & \mathrm{y}_2 & \cdots & \mathrm{y}_\mathrm{n} \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

De esta forma, obtenemos una representación matricial por cada escaneo del láser.

El problema de hallar una transformación rígida que describa cómo se alinean correctamente dos escaneos consecutivos es conocido como método de registración (o *scan matching*)**[10, 15, 16]**. Dos nubes de puntos P y Q se encuentran en una registración cuando para cualquier par de puntos p_i, q_j, cada uno perteneciente a una de las nubes, los mismos representan el mismo objeto en dos escaneos sucesivos. Es decir que existe una transformación rígida $\mathrm{H} \in \mathbb{R}^{3\times3}$ tal que puede alinear todos los puntos de ambas nubes:

$$\forall p_i \in \mathrm{P},\, \forall q_j \in \mathrm{Q},\, \exists \mathrm{H} \,/\, ||\mathrm{H}p_i - q_j|| = 0$$

En general, se trabaja con transformaciones que directamente relacionan las dos nubes de la siguiente forma:

$$\mathrm{H}\, P = Q$$

Entonces, $\mathrm{H} \in \mathbb{R}^{3\times3}$ es la transformación rígida que alinea los puntos de la nube P en la nube Q.

2.4 Método de localización relativa por láser

En el contexto de la localización relativa de un robot móvil que cuenta con un láser, se puede proponer la estimación de su pose a partir de la integración de los movimientos estimados mediante la registración de las nubes de puntos sensadas en cada instante. Es decir, teniendo dos sensados con nubes de puntos asociadas P y Q, si la registración de las mismas permite obtener la transformación rígida H que las relaciona, se puede estimar el movimiento del robot en ese intervalo de tiempo. Así, partiendo de una pose inicial $\mathrm{H}_0 = \mathrm{I}$ (con I, la matriz identidad), la pose en tiempo t del robot se calcula según:

$$\mathrm{H}_t = \mathrm{H}\,\mathrm{H}_{t-1} \ldots \mathrm{H}_0 \tag{2.16}$$

donde H es el movimiento estimado mediante la registración de un par nubes de puntos en tiempo t y $t-1$ respectivamente.

2.5 Iterative Closest Point

El algoritmo ICP introducido por Besl y McKay [17]; Chen y Medion [18] permite calcular una registración entre un par de nubes de puntos. Aproxima de manera iterativa una trasformación rígida para alinear una nube de puntos respecto de la otra. La nube de puntos de referencia se mantiene fija, mientras que la otra nube se transforma para minimizar el error respecto a la referencia. Se refina iterativamente la transformación rígida necesaria para minimizar la distancia desde la nube de puntos de entrada a la nube de puntos de referencia. La entrada del algoritmo son ambas nubes de puntos, un criterio de parada y, opcionalmente, una estimación inicial de la transformación. La salida del algoritmo es una transformación con un error por debajo de un cierto

umbral definido, de tal modo que el resultado obtenido esté sujeto a un criterio de aceptación y, por consiguiente, se asegura un mínimo en cuanto a la calidad de la estimación.

Esencialmente, los pasos del algoritmo ICP son:

1. Para cada punto de la nube de puntos fuente, buscar el punto más cercano en la nube de puntos de referencia y establecer una correspondencia entre ellos (match). En la primera iteración del algoritmo se aplica la estimación inicial (si se contase con ella) de la transformación a la nube fuente.
2. Estimar la transformación que mejor alinee cada punto de la nube fuente con su correspondiente punto en la nube referencia hallado en el paso anterior.
3. Transformar todos los puntos de la nube fuente mediante la transformación obtenida en el paso anterior.
4. Determinar si la transformación obtenida en el paso anterior es lo suficientemente buena usando como función de costo basada en el error (por ejemplo, el error cuadrático medio).
5. Si la transformación obtenida es suficientemente buena, terminar, sino iterar (volver al paso 1).

Para aparear un par de puntos, es decir, establecer correspondencias entre la nube fuente y la nube referencia (paso 1) se realiza la búsqueda del k-ésimo vecino más cercano utilizando un árbol *k-dimensional* o árbol *k-d* [19]

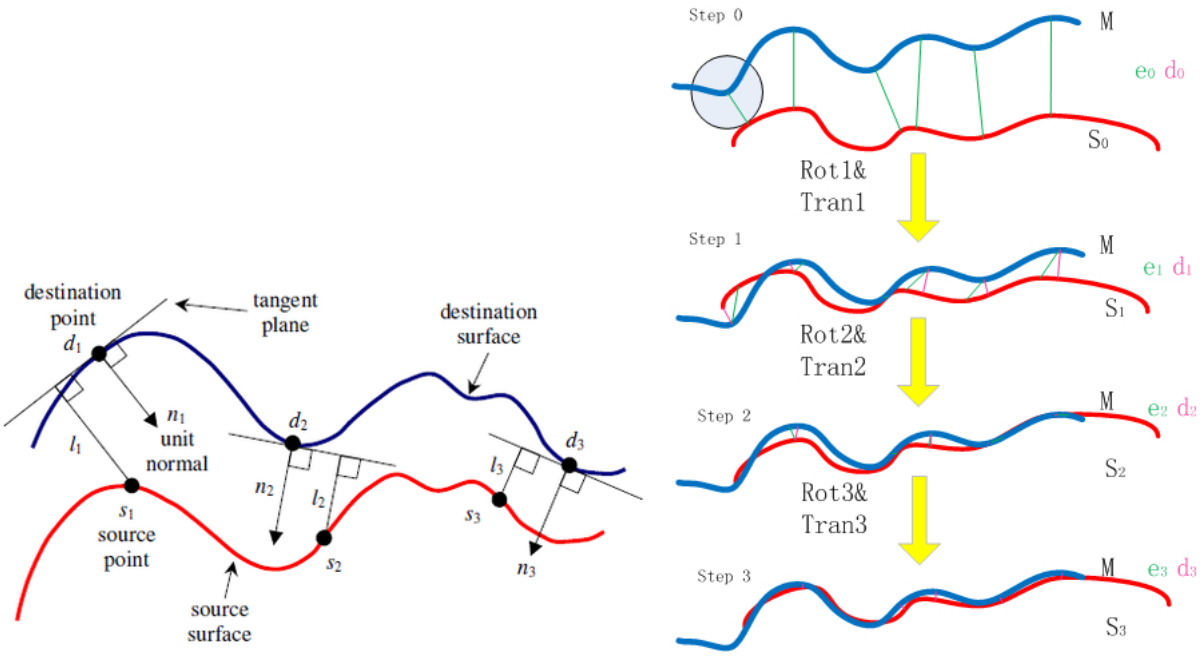
Por otro lado, utilizar todos los puntos muestreados para estimar la transformación (paso 2) impacta tanto en la convergencia del método como en la calidad de la transformación obtenida. Esto se debe a que algunos puntos podrían ser el resultado de errores de medición (ruido), que no sólo no aportan información sino que dificultan la generación de la solución correcta.

Con los pares de puntos ya apareados se construye una transformación $H \in \mathbb{R}^{3x3}$que describe cómo se alinean aproximadamente. Es decir, para $\mathbf{p}_i \in \mathbb{R}^3$y $\mathbf{m}_i \in \mathbb{R}^3$, un punto de la nube fuente y un punto de la nube referencia respectivamente, hay una hay transformación lineal H tal que

$$H\mathbf{p}_i \simeq \mathbf{m}_i$$

Entonces, para determinar cuán buena es la transformación obtenida, se emplea una métrica basada en la sumatoria del error cuadrático medio con respecto a las distancias existentes entre los puntos $\mathbf{m}_i$y los puntos $\mathbf{p}_i$transformados con la transformación H.

Se distinguen dos métricas de error cuadrático medio: point-to-point [17] y point-to-plane[18]. La primera mide el error a partir del cálculo de la distancia de un punto a su vecino más cercano. La segunda, consiste en medir la distancia de un punto al plano definido por los vecinos del punto correspondiente en la otra nube (ver figura 2.4a).



(a) Métrica de error point-to-plane entre dos nubes de puntos (b) Aproximación de las curvas paso a paso

Figura 2.4: La métrica point-to-plane y el ajuste de las curvas a través de las sucesivas iteraciones del método

Formalmente, para point-to-point, la función de error esta dada por

$$E(R,t) = \sum_i \|R\mathbf{p}_i + t - \mathbf{m}_i\|^2$$

donde $R \in \mathbb{R}^{3x3}$ y $t \in \mathbb{R}^3$corresponden a la rotación y traslación asociados a la posición estimada. Esta técnica es generalmente muy estable, pero es bastante lenta y requiere de preprocesamiento, perjudicando la performance a la hora de obtener la registración.

Por otro lado, si utilizamos el método de point-to-plane, tenemos que:

$$E(R,t) = \sum_i \|R\mathbf{p}_i + t - \tilde{\mathbf{m}}_i\|^2$$

donde $R \in \mathbb{R}^{3x3}$ y $t \in \mathbb{R}^3$corresponden a la rotación y traslación asociados a la posición estimada respectivamente, y $\tilde{\mathbf{m}}_i$es la proyección de $\mathbf{p}_i$en el plano correspondiente a $\mathbf{m}_i$.

En cualquiera de los dos casos, la transformación H es considerada como la solución si $E(R,t) \leq \tau$, donde $\tau \in \mathbb{R}^3$es un umbral definido de acuerdo a la fidelidad con la que se requiera la estimación. Lo anterior asegura la convergencia a un mínimo local, partiendo de una estimación inicial, y por lo tanto el algoritmo siempre termina.

2.5.1 Estimación inicial de ICP

En métodos iterativos de registración como ICP, es común pasar como parámetro una estimación inicial de la transformación, que sirve como punto de partida. Hacer uso de una buena estimación

inicial contribuye a disminuir la probabilidad de converger a mínimos locales no globales y a reducir la cantidad de iteraciones necesarias para llegar a una solución. La estimación inicial puede obtenerse a partir de la información que ofrece alguno de los otros sensores montados en el robot que no sea el utilizado para generar las nubes de puntos. En el caso de robots terrestres cuyo sistema de locomoción está basado en ruedas, la odometría basada en encoders suele cumplir este rol.

2.5.2 Registración del láser mediante keyframes

Como fue mencionado en 2.3, cada vez que se recibe un nuevo escaneo por láser se construye la nube de puntos asociada y se calcula la registración utilizando el método iterativo ICP. Cada nueva registración implica una actualización respecto de la trayectoria realizada por el robot hasta el momento, sin embargo, no tiene sentido incorporar mediciones que no contengan nueva información e introduzcan ruido, por lo que seleccionaremos sólo las mediciones relevantes.

La detección de muestras relevantes (o *keyframes)* se basa en establecer un criterio de selección sobre qué transformaciones deben ser acumuladas. Siempre que se considere que existe un cambio significativo a partir del *keyframe* anterior, éste pasa a ser el nuevo *keyframe scan*.

En el presente trabajo, la métrica para decidir cuándo el muestreo es un *keyframe* utiliza los valores proporcionados por la odometría respecto de cuánto se ha desplazado el robot respecto del último *keyframe*. Para ello se acumula el desplazamiento lineal Δd y angular $\Delta\theta$ observado en este ultimo intervalo de tiempo

$$\begin{aligned} \Delta d_t &= \Delta d_{t-1} + |\Delta d| \\ \Delta\theta_t &= \Delta\theta_{t-1} + \Delta\theta \end{aligned}$$

Un nuevo sensado será considerado *keyframe* si excede ciertos umbrales μ_θ, μ_d. Estos umbrales deberán ser obtenidos experimentalmente para cada caso con cuidado de no elegir valores demasiado grandes ya que resultaría en *keyframes* muy espaciados. Esto resultaría en nubes de puntos muy diferentes entre si y en una mayor dificultad al momento de realizar la registración entre ambas.

El correcto uso de *keyframes* ayuda a minimizar la acumulación del error (inducido por el ruido propio del sensado) y, por lo tanto, contribuye a producir mejores estimaciones de la pose del robot en cada momento.

2.6 Fusión de sensores

Tanto los encoders asociados a las ruedas como el telémetro láser son sensores que proveen información a partir de la cuál es posible realizar una estimación del movimiento del robot. Sin embargo, esta estimación suele estar acompañada de errores debido a limitaciones físicas de los sensores. Por ejemplo, la odometría basada en encoders no suele ser lo suficientemente buena para estimar movimientos de rotación pero sí de traslación, debido a que al girar el robot sobre su propio eje las ruedas tienden a patinar y esto genera un error no sistemático de odometría. Por otro lado, el telémetro láser puede resultar impreciso en recorridos dónde los escaneos no detectan cambios significativos en el ambiente, por ejemplo, al avanzar sobre un pasillo recto suficientemente largo y uniforme. Sin embargo, en la mayoría de los casos existe suficiente información para estimar en forma precisa la rotación.

Puesto que cada nueva medición de los sensores introduce un nuevo error y éste es acumulado mediante las transformaciones resultantes, el error en la estimación crece indefinidamente. Para abordar este problema, se puede fusionar [20] la información provista por distintos sensores de manera complementar las limitaciones de cada uno. Es decir, elaborar un mecanismo que integre en la estimación lo "mejor" de cada sensor. Existen diversos métodos para fusionar la información

proveniente de múltiples fuentes, siendo los más utilizados el filtro complementario [21, 22] o el Filtro de Kalman [23]. En particular, en el presente trabajo se utiliza un Filtro Extendido de Kalman.

2.7 Filtro de Kalman

El Filtro de Kalman, propuesto por Rudolf E. Kalman en 1960 [24], es un algoritmo para identificar el estado oculto de un sistema dinámico lineal que no puede ser medido de forma directa. Es un estimador recursivo, es decir que únicamente son necesarias la estimación de un estado anterior y un modelo de proceso para poder generar una predicción para el estado actual. En contraste con las técnicas de estimación *batch* no es necesario contar con un historial de observaciones y estimaciones.

En robótica, el Filtro de Kalman es ampliamente utilizado para resolver problemas de localización, mapeo y navegación.

Si bien el Filtro de Kalman puede ser definido utilizando una única ecuación, es común distinguir dos fases:

- La fase de *predicción*
- La fase de *actualización* (o corrección)

La fase de predicción consiste en utilizar un *modelo de proceso*, que en el contexto de robótica móvil también es llamado *modelo de movimiento*, para estimar el estado en el tiempo actual a partir de un estado en un tiempo anterior. El mismo se denomina estado *a priori* y se denota:

$$\mathbf{x}_{k|k-1} \in \mathbb{R}^{\mathrm{n}}$$

donde $\mathbf{x}_{k|k-1}$ es una variable Gausiana y expresa la predicción del estado en el instante k utilizando la información del instante $k-1$.

Por otro lado, la fase de actualización se basa en combinar el estado a priori junto con una *observación* (o medición) actual, total o parcial, que es obtenida por alguna fuente que provea los datos. Esta actualización es conocida como estado *a posteriori* y se denota:

$$\mathbf{x}_{k|k} \in \mathbb{R}^{\mathrm{n}}$$

donde $\mathbf{x}_{k|k}$ expresa el estado en el instante k, utilizando las observaciones hasta ese mismo instante.

Para poder calcular tanto el estado a priori como el estado a posteriori, el Filtro de Kalman requiere dos modelos asociados al contexto del problema:

- Modelo de movimiento
- Modelo de medición (u observación)

El modelo de movimiento describe cómo evoluciona el sistema a lo largo del tiempo. En el caso de un robot móvil, describe el movimiento físico del robot en su recorrido. El modelo de observación establece cómo se incorpora la información de las mediciones obtenidas por alguna fuente de información a la actualización del estado.

2.7.1 Predicción del estado

En la fase de predicción se obtiene el estado a priori $\mathbf{x}_{k|k-1} \in \mathbb{R}^{\mathrm{n}}$ según

$$\mathbf{x}_{k|k-1} = \mathrm{F}_k \, \mathbf{x}_{k-1|k-1} + \ w_k$$

donde $\mathrm{F}_k \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ representa el modelo de movimiento y $\mathrm{w_k} \sim \mathcal{N}(0,\ \mathrm{Q_k})$ modela el error de dicho modelo en forma de ruido blanco gaussiano. La matriz $\mathrm{Q}_k \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ denota la covarianza del error w_k.

Por otra parte, la covarianza asociada a dicho estado, denotada $\mathrm{P}_{k|k-1}\in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$, se obtiene del siguiente modo,

$$\mathrm{P}_{k|k-1} = \mathrm{F_k} \, \mathrm{P}_{k-1|k-1} \, \mathrm{F}_k^t + \mathrm{Q_k}$$

2.7.2 Actualización del estado

Para llevar a cabo la fase de actualización se debe definir el modelo de observación $\mathrm{H_k} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ que relaciona el valor obtenido por el sensor $\mathbf{z}_k \in \mathbb{R}^{\mathrm{n}}$ con el estado predicho $\mathbf{x}_{k|k-1} \in \mathbb{R}^{\mathrm{n}}$, según

$$\mathbf{z}_k \simeq \mathrm{H}_k \, \mathbf{x}_{k|k-1} + \mathbf{v}_k$$

donde $\mathbf{v}_k \sim \mathcal{N}(0,\ \mathrm{R}_k) \in \mathbb{R}^{\mathrm{n}}$ modela el ruido de medición como ruido blanco gaussiano. Llamamos $\mathrm{R}_k\in\mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ la matriz de covarianza asociada al mismo. Los cálculos involucrados para realizar la actualización, es decir, obtener el estado a *posteriori* son los siguientes:

$$\mathbf{y}_k = \mathbf{z}_k - \mathrm{H}_k \, \mathbf{x}_{k|k-1}$$

$$\mathrm{S}_k = \mathrm{H}_k \, \mathrm{P}_{k|k-1} \, \mathrm{H}_k^t + \mathrm{R}_k$$

$$\mathrm{K}_k = \mathrm{P}_{k|k-1} \, \mathrm{H}_k^t \, \mathrm{S}_k^{-1}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathrm{K}_k \, \mathbf{y}_k$$

$$\mathrm{P}_{k|k} = (\mathrm{I} - \mathrm{K}_k \, \mathrm{H}_k) \, \mathrm{P}_{k|k-1}$$

donde $\mathbf{y}_k \in \mathbb{R}^{\mathrm{n}}$ corresponde a la diferencia entre el sensado real $\mathbf{z}_k$ y el estado a priori, $\mathbf{x}_{k|k-1}$, transformado por el modelo de medición, conocido como innovación o medición residual. Por otro lado, el estimado $\mathrm{K}_k \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$ corresponde a la denominada matriz de ganancias, que expresa como impacta la diferencia $\mathbf{y}_k$ en una corrección sobre la estimación.

2.7.3 Filtro de Kalman Extendido

La versión extendida del Filtro de Kalman es una extensión del método original que permite aplicarlo a sistemas donde los modelos de movimiento y medición pueden ser no-lineales. En este caso, se definen ambos modelos mediante una función f y h , respectivamente, según

$$\mathbf{x}_k = f\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) + \mathbf{w}_k$$

$$\mathbf{z}_k = h\left(\mathbf{x}_k\right) + \mathbf{v}_k$$

donde $\mathbf{w}_k \sim \mathcal{N}(0,\, \mathrm{Q}_{k-1})$ y $\mathbf{v}_k \sim \mathcal{N}(0,\, \mathrm{R}_k)$ son los ruidos asociados a cada modelo y $\mathbf{u}_{k-1} \in \mathbb{R}^{\mathrm{n}}$ es una entrada al modelo de movimiento (conocido como vector de control).

Para obtener el Filtro de Kalman Extendido, se procede a linealizar dichas funciones utilizando una aproximación de Taylor de primer orden, a partir del cálculo de los jacobianos $\widehat{\mathrm{F}}$ y $\widehat{\mathrm{H}}$ asociados a f y h, respectivamente:

$$\widehat{\mathrm{F}}_{k-1} = \left.\frac{\partial \mathrm{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{k-1|k-1}, \mathbf{u}_{k-1}}$$

$$\widehat{\mathrm{H}}_k = \left.\frac{\partial \mathrm{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{k|k-1}}$$

Siendo evaluado tanto $\mathbf{x}_{k-1|k-1}$ y $\mathbf{u}_{k-1}$ el jacobiano de la función f y, únicamente por el primero para el jacobiano de la función h.

El paso de predicción, entonces, resulta:

$$\mathbf{x}_{k|k-1} = f\left(\mathbf{x}_{k-1|k-1},\, \mathbf{u}_{k-1}\right)$$

$$\mathrm{P}_{k|k-1} = \widehat{\mathrm{F}}_{k-1}\, \mathrm{P}_{k-1|k-1}\, \widehat{\mathrm{F}}_{k-1}^{t} + \mathrm{Q}_{k-1}$$

Por otro lado, la fase de actualización se define de la siguiente forma:

$$\mathbf{y}_k = \mathbf{z}_k - h\left(\mathbf{x}_{k|k-1}\right)$$

$$\mathrm{S_k} = \widehat{\mathrm{H}}_k\, \mathrm{P}_{k|k-1}\, \widehat{\mathrm{H}}_k^{t} + \mathrm{R}_k$$

$$\mathrm{K}_k = \mathrm{P}_{k|k-1}\, \widehat{\mathrm{H}}_k^{t}\, \mathrm{S}_k^{-1}$$

$$\mathrm{P}_{k|k} = \left(\mathrm{I} - \mathrm{K}_k\, \widehat{\mathrm{H}}_k\right) \mathrm{P}_{k|k-1}$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathrm{K}_k\, \mathbf{y}_k$$

Las ecuaciones y, en consecuencia, los cálculos involucradas en el Filtro de Kalman extendido únicamente difieren en la utilización de los jacobianos de las funciones asociadas a los modelos en lugar de utilizarlos directamente.

2.7.4 Aplicación en robótica móvil

En el caso del problema de localización para un robot móvil, el estado representa la pose del robot en un momento dado. Dado un robot que trabaja sobre un plano en dos dimensiones podemos definir el vector de estado para el instante k como $\mathbf{x}_k = [x_k, y_k, \theta_k, v_{x,k}, v_{y,k}, v_{\theta,k}]$, donde x e y

representan la posición del robot, θ la orientación en el plano y v_x, v_y, v_θ las velocidades en el plano y angular, respectivamente.

En el presente trabajo, se utiliza un modelo cinemático basado en mecánica Newtoniana clásica, definido por la siguiente función:

$$f_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \\ v_{x,k-1} \\ v_{y,k-1} \\ v_{\theta,k-1} \end{bmatrix} + \begin{bmatrix} \Delta_t \left(v_{x,k-1} \cos(\theta_{k-1}) - v_{y,k-1} \sin(\theta_{k-1}) \right) \\ \Delta_t \left(v_{x,k-1} \sin(\theta_{k-1}) + v_{y,k-1} \cos(\theta_{k-1}) \right) \\ \Delta_t v_{\theta,k-1} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Este es un modelo cinemático de un robot omnidireccional (holonómico) con velocidades constantes, aunque luego se realizaron experimentos con un robot con tracción diferencial (no holonómico). Es decir, con mayores restricciones en el modelo, en particular $v_y = 0$ ya que este movimiento no es posible en un sistema de tracción diferencial.

El error del modelo de movimiento debe ajustarse experimentalmente, y estará representado por su matriz de covarianza Q_k.

A su vez, debemos definir el modelo de observación h_k, que dependerá de como se relacionen las mediciones $\boldsymbol{z}_k$ con el vector de estado. En este trabajo, tanto la odometría basada en encoders como la registración por ICP proporcionan una pose relativa a un instante previo, que modelamos como velocidades (v_x, v_y, v_θ), de manera que definimos h como

$$h_k(\mathbf{x}_k) = \begin{bmatrix} v_{x,k} \\ v_{y,k} \\ v_{\theta,k} \end{bmatrix}$$

Para poder realizar con precisión la fase de actualización, resulta necesario caracterizar las incertezas en las nuevas mediciones $\boldsymbol{z}_k$ que provienen de cada fuente de información. En nuestro caso, esto implica conocer las matrices de covarianza R_k para las mediciones resultantes tanto de la localización relativa por odometría (2.1), como para la localización relativa por láser (2.4). Estas incertezas no provienen solamente del error producido en los sensores, sino también de los algoritmos utilizados para procesar la información provenientes de los mismos, como ser la registración por ICP, o la integración de la odometría.

Capítulo 3

Método para el cálculo de la covarianza de ICP

Para obtener información sobre el resultado del algoritmo ICP y poder fusionar esta información con la de odometría mediante EKF, se desea estimar el error producido al calcular una registración del láser. Para esto, es necesario calcular una matriz de covarianza junto con el resultado de la registración de cada par de nubes de puntos. La matriz de covarianza provee información sobre la precisión del resultado obtenido por el algoritmo. El resultado de la registración de un par de nubes de puntos para el caso 2D es una transformación $\mathrm{R} \in \mathbb{R}^{3\mathrm{x}3}$ y esta transformación tiene 3 parámetros $(\mathrm{x}, \mathrm{y}, \theta)$.

Para calcular esta matriz se analizaron dos enfoques: El cálculo de la covarianza por muestreo del espacio de resultados y el calculo por una fórmula cerrada [25, 26].

3.1 Muestreo del espacio de resultados

Este método consiste en el muestreo de todos los posibles resultados del algoritmo, para así obtener una caracterización de la distribución de probabilidad de los resultados. Este proceso es computacionalmente muy costoso, por lo que resulta impracticable para aplicaciones de tiempo real.

Un algoritmo que implementa este enfoque puede ser el siguiente:

1. Almacenar una lectura completa del láser
2. Repetir los siguientes pasos n veces, donde n es el numero de muestreos a realizar:
   (a) Generar un desplazamiento al azar $\bar{\mathrm{x}} = [\mathrm{x}, \mathrm{y}, \theta]$.
   (b) Simular una nueva nube de puntos basada en la nube original y el desplazamiento $\bar{\mathrm{x}}$.
   (c) Ejecutar el algoritmo de ICP entre estas dos nubes de puntos, y almacenar el error $(\hat{\mathrm{x}} - \bar{\mathrm{x}})$.
3. Calcular la matriz de covarianza en base a los errores de registración,. i.e. $Cov\left(\hat{\mathrm{x}}\right) = E\left[\left(\hat{\mathrm{x}} - \bar{\mathrm{x}}\right)^T \left(\hat{\mathrm{x}} - \bar{\mathrm{x}}\right)\right] = \left(\hat{\mathrm{x}} - \bar{\mathrm{x}}\right)^T \left(\hat{\mathrm{x}} - \bar{\mathrm{x}}\right)/n$

Este algoritmo no resulta aplicable en la practica, debido a la cantidad de nubes simuladas que deben ser calculadas y registradas mediante ICP. Además, la simulación de los desplazamientos no se puede hacer con precisión, ya que se desconoce el mapa real del entorno.

3.2 Forma cerrada para la covarianza de un algoritmo de minimización

En contraposición al enfoque basado en muestreo, Bengtsson et al. [26] propone el llamado *Método del Hessiano*. Este método es rápido de calcular, pero no es suficientemente preciso. Por otro lado, Censi [25] propone un método más preciso para calcular la covarianza de un algoritmo de minimización. El algoritmo en este caso es el que utiliza ICP para minimizar la función de error $J(\mathrm{x},\mathrm{z})$ y cuyo resultado es una transformación $\hat{\mathrm{x}}$. La función de error $J(\mathrm{x},\mathrm{z})$ dependerá de la implementación de ICP que estemos utilizando, según vimos en 2.5. En este caso, utilizaremos el método *point-to-plane* para medir el error.

En [27] se deriva la siguiente función de error para la versión *point-to-plane* de ICP:

$$J(\rho_q,\rho_p,x,y,\theta)=\sum_k\left(\left(\begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\cdot\rho_{q^k}\overrightarrow{v}_{q^k}+\begin{bmatrix}x\\ y\end{bmatrix}-\rho_{p^l}\vec{v_{p^l}}\right)\vec{n_{p^l}}\right)^2 \tag{3.1}$$

donde l es el índice del punto p en la lectura de referencia que esta a menor distancia del punto $q_k$en la nueva lectura.

Entonces, si llamamos $\mathbf{A}$ al algoritmo de ICP, y $\hat{\mathrm{x}}$ al resultado del mismo, y sabemos que ICP trabaja minimizando el error de una función $J(\mathrm{x},\check{\mathrm{z}})$, donde $\check{z}$ representa las nubes de puntos obtenidas por el láser, podemos escribirlo de la siguiente forma:

$$\hat{\mathrm{x}}=\mathbf{A}(\check{z})=\arg\min_x J(\check{z},x)$$

La aproximación de primer orden a la covarianza de $\hat{\mathrm{x}}$, esta dada entonces por:

$$Cov(\hat{\mathrm{x}})\simeq\frac{\partial\mathbf{A}}{\partial z}\,Cov(z)\,\frac{\partial\mathbf{A}}{\partial z}^T \tag{3.2}$$

Como no conocemos una forma cerrada de A, no es sencillo calcular $\partial A/\partial z$. Sin embargo, $A(z)$ y z están relacionadas por una función implícita. De hecho, $\hat{\mathrm{x}}$ es punto critico de J. $\partial J(\check{z},x)/\partial x=0$.

El teorema de la función implícita nos da una expresión para $\partial A/\partial z$. Aplicando el teorema con $F=\partial J/\partial x$,$f(z)=A(z)$,$x_0=x$, obtenemos:

$$\left.\frac{\partial\mathbf{A}(z)}{\partial z}\right|_{z=\check{z}}=-\left(\frac{\partial^2 J}{\partial x^2}\right)^{-1}\left.\frac{\partial^2 J}{\partial z\partial x}\right|_{x=\mathbf{A}(\check{z})}$$

Reemplazando esta ultima ecuación en 3.2, obtenemos:

$$Cov(\hat{\mathrm{x}})\cong\left(\frac{\partial^2 J}{\partial \mathrm{x}^2}\right)^{-1}\frac{\partial^2 J}{\partial \mathrm{x}\partial \mathrm{z}}\,Cov(\mathrm{z})\,\frac{\partial^2 J}{\partial \mathrm{x}\partial \mathrm{z}}^T\left(\frac{\partial^2 J}{\partial \mathrm{x}^2}\right)^{-1} \tag{3.3}$$

La forma cerrada de la covarianza se calcula entonces utilizando la función de error $J(\mathrm{x},\mathrm{z})$ del paso de minimización de ICP (3.1). En esta función de error, cada punto q_k de la lectura del láser es transformado a su forma polar $q_k=\rho_k\cdot\overrightarrow{v_k}$ para calcular la covarianza. En la forma polar, cada haz del sensor tiene una probabilidad en una sola dirección $\overrightarrow{v_k}$ con una varianza de σ_ρ^2. Suponiendo σ^ρ constante, la matriz de covarianza z del sensor resulta diagonal ($cov(\mathrm{z})=diag\left(\sigma_\rho^2\right)$) y por lo tanto podemos simplificar σ_ρ^2 multiplicando adelante de la ecuación. Además, por ser $\left(\frac{\partial^2 J}{\partial \mathrm{x}^2}\right)^{-1}$ una matriz simétrica, podemos reescribir la ecuación anterior de la siguiente manera:

$$Cov\left(\hat{\mathrm{x}}\right) \cong \sigma_\rho^2 \frac{\partial A}{\partial \mathrm{z}} \cdot \frac{\partial A}{\partial \mathrm{z}}^T \tag{3.4}$$

con

$$\frac{\partial A}{\partial \mathrm{z}} = \left(\frac{\partial^2 J}{\partial \mathrm{x}^2}\right)^{-1} \frac{\partial^2 J}{\partial \mathrm{x} \partial \mathrm{z}} \tag{3.5}$$

Las segundas derivadas de $J\left(\mathrm{x}, \mathrm{z}\right)$ que son necesarias para calcular la covarianza resultan entonces de la siguiente forma:

$$\frac{\partial^2 J}{\partial \mathrm{x}^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial x^2} & \frac{\partial^2 J}{\partial x \partial y} & \frac{\partial^2 J}{\partial x \partial \theta} \\ \frac{\partial^2 J}{\partial x \partial y} & \frac{\partial^2 J}{\partial y^2} & \frac{\partial^2 J}{\partial y \partial \theta} \\ \frac{\partial^2 J}{\partial x \partial \theta} & \frac{\partial^2 J}{\partial y \partial \theta} & \frac{\partial^2 J}{\partial \theta^2} \end{bmatrix}$$

$$\frac{\partial^2 J}{\partial \mathrm{x} \partial \mathrm{z}} = \begin{bmatrix} \frac{\partial^2 J}{\partial x \partial \rho_{q_1}} & \cdots & \frac{\partial^2 J}{\partial x \partial \rho_{q_k}} & \frac{\partial^2 J}{\partial x \partial \rho_{p_1}} & \cdots & \frac{\partial^2 J}{\partial x \partial \rho_{p_k}} \\ \frac{\partial^2 J}{\partial y \partial \rho_{q_1}} & \cdots\cdots & \frac{\partial^2 J}{\partial y \partial \rho_{q_k}} & \frac{\partial^2 J}{\partial y \partial \rho_{p_1}} & \cdots\cdots & \frac{\partial^2 J}{\partial y \partial \rho_{p_k}} \\ \frac{\partial^2 J}{\partial \theta \partial \rho_{q_1}} & \cdots\cdots & \frac{\partial^2 J}{\partial \theta \partial \rho_{q_k}} & \frac{\partial^2 J}{\partial \theta \partial \rho_{p_1}} & \cdots\cdots & \frac{\partial^2 J}{\partial \theta \partial \rho_{p_k}} \end{bmatrix}$$

Las derivadas necesarias para calcular los jacobianos son extensas, pero podemos obtener una forma cerrada sencillamente. Primero reescribiremos la función J sin formas matriciales:

$$\begin{aligned} J\left(\rho_q, \rho_p, x, y, \theta\right) &= \sum_k \left(\left(\begin{bmatrix} \cos\left(\theta\right) & -\sin\left(\theta\right) \\ \sin\left(\theta\right) & \cos\left(\theta\right) \end{bmatrix} \cdot \rho_{q^k} \overrightarrow{v}_{q^k} + \begin{bmatrix} x \\ y \end{bmatrix} - \rho_{p^l} \vec{v_{p^l}} \right) \vec{n_{p^l}} \right)^2 \\ &= \sum_k \big(\cos\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l x} - \sin\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l x} + x\, n_{p^l x} - v_{p^l x} \rho_{p^l} n_{p^l x} \\ &\qquad + \sin\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l y} + \cos\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l y} + y\, n_{p^l y} - v_{p^l y} \rho_{p^l} n_{p^l y} \big)^2 \end{aligned}$$

Luego, debemos calcular las derivadas para $\frac{\partial J}{\partial \mathrm{x}}$:

$$\frac{\partial J}{\partial \mathrm{x}} = \begin{bmatrix} \frac{\partial J}{\partial x} \\ \frac{\partial J}{\partial y} \\ \frac{\partial J}{\partial \theta} \end{bmatrix}$$

$$\begin{aligned} \frac{\partial J}{\partial x} = \sum_k 2 n_{p^l x} \big(&\cos\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l x} - \sin\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l x} + x\, n_{p^l x} - v_{p^l x} \rho_{p^l} n_{p^l x} \\ &+ \sin\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l y} + \cos\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l y} + y\, n_{p^l y} - v_{p^l y} \rho_{p^l} n_{p^l y} \big) \end{aligned}$$

$$\begin{aligned} \frac{\partial J}{\partial y} = \sum_k 2 n_{p^l y} \big(&\cos\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l x} - \sin\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l x} + x\, n_{p^l x} - v_{p^l x} \rho_{p^l} n_{p^l x} \\ &+ \sin\left(\theta\right) v_{q^k x} \rho_{q^k} n_{p^l y} + \cos\left(\theta\right) v_{q^k y} \rho_{q^k} n_{p^l y} + y\, n_{p^l y} - v_{p^l y} \rho_{p^l} n_{p^l y} \big) \end{aligned}$$

$$\frac{\partial J}{\partial \theta} = \sum_k \big(2\left(\cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x} + x\, n_{p^l x} - v_{p^l x}\rho_{p^l} n_{p^l x}\right.$$
$$\left. + \sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y} + \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y} + y\, n_{p^l y} - v_{p^l y}\rho_{p^l} n_{p^l y}\right)$$
$$\left(-\sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} - \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x} + \cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y}\right)\big)$$

A continuación, utilizando el resultado anterior, proseguiremos con las derivadas para $\frac{\partial^2 J}{\partial \mathrm{x}^2}$:

$$\frac{\partial^2 J}{\partial x^2} = \sum_k 2 n_{p^l x}^2$$

$$\frac{\partial^2 J}{\partial y^2} = \sum_k 2 n_{p^l y}^2$$

$$\frac{\partial^2 J}{\partial x \partial y} = \frac{\partial^2 J}{\partial y \partial x} = \sum_k 2 n_{p^l x} n_{p^l y}$$

$$\frac{\partial^2 J}{\partial x \partial \theta} = \frac{\partial^2 J}{\partial \theta \partial x} = \sum_k 2 n_{p^l x}\left(-\sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} - \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x}\right.$$
$$\left. + \cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y}\right)$$

$$\frac{\partial^2 J}{\partial y \partial \theta} = \frac{\partial^2 J}{\partial \theta \partial y} = \sum_k 2 n_{p^l y}\left(-\sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} - \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x}\right.$$
$$\left. + \cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y}\right)$$

$$\frac{\partial^2 J}{\partial \theta^2} = \sum_k 2\Big(\left(\cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x} - \cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y} + \sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} + \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y}\right)^2$$
$$- \left(\cos(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} + \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x} + \sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y}\right)$$
$$\left(n_{p^l x}\, x + n_{p^l y}\, y - v_{p^l x}\rho_{p^l} n_{p^l x} - v_{p^l y}\rho_{p^l} n_{p^l y}\right.$$
$$\left. + \cos(\theta)\,(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l x} + \cos(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l y} - \sin(\theta)\, v_{q^k y}\rho_{q^k} n_{p^l x} + \sin(\theta)\, v_{q^k x}\rho_{q^k} n_{p^l y}\right)\Big)$$

Finalmente, calcularemos las derivadas para $\frac{\partial^2 J}{\partial \mathrm{x} \partial \mathrm{z}}$:

$$\frac{\partial^2 J}{\partial x \partial \rho_{q^i}} = 2 n_{p^l x}\left(\cos(\theta)\, v_{q^i x} n_{p^l x} - \sin(\theta)\, v_{q^i y} n_{p^l x} + \sin(\theta)\, v_{q^i x} n_{p^l y} + \cos(\theta)\, v_{q^i y} n_{p^l y}\right)$$

$$\frac{\partial^2 J}{\partial y \partial \rho_{q^i}} = 2 n_{p^l y}\left(\cos(\theta)\, v_{q^i x} n_{p^l x} - \sin(\theta)\, v_{q^i y} n_{p^l x} + \sin(\theta)\, v_{q^i x} n_{p^l y} + \cos(\theta)\, v_{q^i y} n_{p^l y}\right)$$

$$\frac{\partial^2 J}{\partial x \partial \rho_{p^i}} = -2n_{p^l x}\left(v_{p^l x}n_{p^l x} + v_{p^l y}n_{p^l y}\right)$$

$$\frac{\partial^2 J}{\partial y \partial \rho_{p^i}} = -2n_{p^l x}\left(v_{p^l x}n_{p^l x} + v_{p^l y}n_{p^l y}\right)$$

$$\begin{aligned}\frac{\partial^2 J}{\partial \theta \partial \rho_{q^i}} = {} & 4\rho_{q^i}\left(\cos(\theta)\, v_{q^i y}n_{p^l x} - \cos(\theta)\, v_{q^i x}n_{p^l y} + \sin(\theta)\, v_{q^i x}n_{p^l x} + \sin(\theta)\, v_{q^i y}n_{p^l y}\right)^2 - \\ & 2\left(\cos(\theta)\, v_{q^i x}n_{p^l x} - \sin(\theta)\, v_{q^i y}n_{p^l x} + \sin(\theta)\, v_{q^i x}n_{p^l y} + \cos(\theta)\, v_{q^i y}n_{p^l y}\right) \\ & \left(\cos(\theta)\, v_{q^i x}\rho_{q^i}n_{p^l x} - \sin(\theta)\, v_{q^i y}\rho_{q^i}n_{p^l x} + x\, n_{p^l x} - v_{p^l x}\rho_{p^l}n_{p^l x}\right. \\ & \left. + \sin(\theta)\, v_{q^i x}\rho_{q^i}n_{p^l y} + \cos(\theta)\, v_{q^i y}\rho_{q^i}n_{p^l y} + y\, n_{p^l y} - v_{p^l y}\rho_{p^l}n_{p^l y}\right) \\ & - 2\rho_{q^i}\left(\cos(\theta)\, v_{q^i x}n_{p^l x} + \cos(\theta)\, v_{q^i y}n_{p^l y} - \sin(\theta)\, v_{q^i y}n_{p^l x} + \sin(\theta)\, v_{q^i x}n_{p^l y}\right)^2\end{aligned}$$

$$\frac{\partial^2 J}{\partial \theta \partial \rho_{p^i}} = 2\rho_{q^i}\left(v_{p^l x}n_{p^l x} + v_{p^l y}n_{p^l y}\right)\left(\cos(\theta)\, v_{p^l x}n_{p^l x} + \cos(\theta)\, v_{p^l y}n_{p^l y} - \sin(\theta)\, v_{p^l y}n_{p^l x} + \sin(\theta)\, v_{p^l x}n_{p^l y}\right)$$

Utilizando estos resultados, es posible calcular la ecuación 3.4 de forma cerrada y de esta forma obtener la covarianza del error para un resultado de ICP, que es lo que queríamos calcular.

Capítulo 4

Implementación

A fines de comprender el sistema de localización propuesto en su totalidad se deben dar a conocer sus detalles de implementación. La solución presentada fue llevada a cabo utilizando tres componentes de software: el framework `ROS` [1] (Robot Operating System), del cual utilizamos diversos componentes, la biblioteca `libpointmatcher` [2] que fue adaptada para su uso con ROS, y nodos adicionales de ROS que fueron desarrollados para poder evaluar los resultados obtenidos.

El framework ROS ofrece una capa de abstracción entre los dispositivos de hardware (como los sensores montados en el robot) y el software (como el algoritmo de EKF). Además resuelve la comunicación con el robot y entre los distintos componentes de software.

La biblioteca `libpointmatcher` permite calcular la registración existente (transformación rígida) entre un par de nubes de puntos, empleando el algoritmo de ICP. Al elegir un algoritmo de *scan-matching* para integrar con el cálculo de covarianza, se tuvieron en cuenta los siguientes requerimientos: posibilidad de aplicar filtros, diferentes métricas de error (point-to-point o point-to-plane), configuración del umbral de convergencia, etc. `libpointmatcher` cumple con lo anterior y, además ofrece una implementación modular que permite integrar fácilmente el algoritmo de cálculo de covarianza.

La utilización de un algoritmo de EKF se basó en el hecho de que se conocía de antemano tanto las bondades que tenían los sensores empleados como así también sus limitaciones. Por un lado, se sabe que el láser es muy preciso cuando el entorno provee suficientes alteraciones (generalmente al realizar un movimiento rotacional) pero si se "mantiene constante" (por ejemplo el recorrido de un pasillo recto, es decir, un desplazamiento traslacional) no proporcionará al algoritmo de ICP con información que permita estimar el movimiento ocurrido cuando realmente sí sucede. Por otro lado, es sabido que la odometría por encoders es muy buena como indicador de movimiento a corto plazo y muy precisa para desplazamientos lineales (no así para rotacionales debido al drift que podría existir en la ruedas, diferencias en la potencia del motor asociado a cada una, etc.). La motivación de utilizar un método de fusión de sensores, se centra en explotar las fortalezas de cada sensor (y su consiguiente estimación de la posición) de modo de conllevar a estimaciones más robustas, menos sensibles de verse afectadas por ciertas características del entorno, y con mayor precisión. La fusión de la información provista por ICP con la información provista por odometría es llevada a cabo mediante un módulo de `ROS` llamado `robot_localization [28]`, que implementa el algoritmo de EKF.

[1] http://www.ros.org/

[2] https://github.com/ethz-asl/libpointmatcher

4.1 Biblioteca de registración de sensado

Para calcular la registración existente entre un par de nubes de puntos se utilizó la biblioteca `libpointmatcher` [29] que implementa el algoritmo de ICP. Para utilizar esta biblioteca se requiere rellenar una serie de estructuras de datos de entrada (para representar las nubes de puntos, la estimación inicial del algoritmo ICP, etc.) y manipular otra estructura de datos de salida para obtener la transformación deseada (registración).

El proceso de la registración se puede dividir en las siguientes etapas:

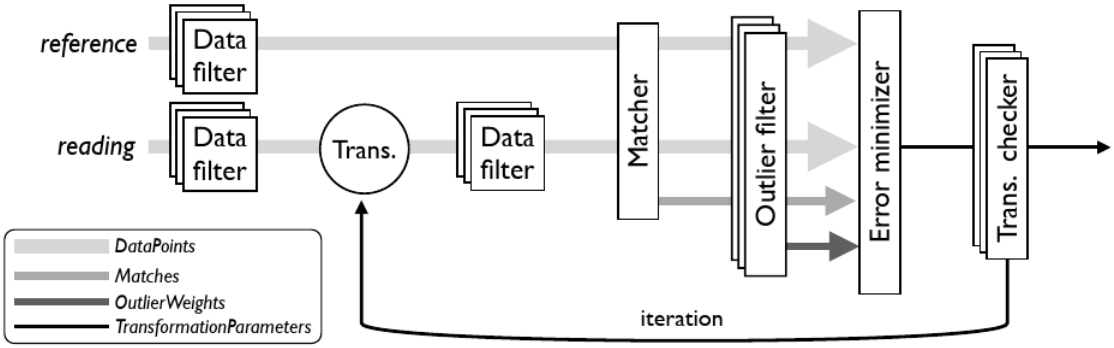


Figura 4.1: Etapas del proceso para obtener la registración mediante ICP.

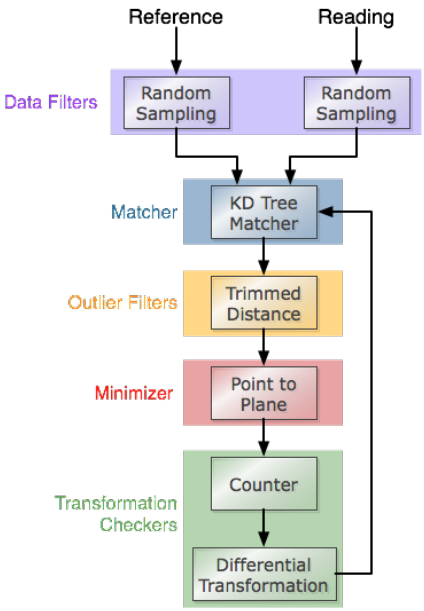


Figura 4.2: Configuración por defecto para la utilización de libpointmatcher

En primer lugar, ambas nubes de puntos se les aplica una serie de *filtros* con la intención de mitigar ciertos factores del ambiente (por ejemplo eliminar puntos que no son de interés o que pueden ser espurios debido al ruido del sensor). Luego, los puntos de la nube fuente son transformados (rotados y trasladados) utilizando la transformación estimada hasta el momento (en el caso de la primer iteración utilizando la estimación inicial calculada por odometría). Se establecen correspondencias utilizando una árbol *k-d y* el criterio del *k*-ésimo vecino más cercano. Como tercer paso se realiza una eliminación de outliers y se pondera las correspondencias establecidas asignándoles diferentes pesos. Se pueden utilizar distintos criterios, como por ejemplo, fijar una distancia máxima, un factor respecto de la distancia media, etc. Luego, se calcula la transformación que minimiza el error entre ambas nubes de puntos pudiéndose aplicar dos diferentes métricas: point-to-point o point-to-plane. Por último, se verifica si el criterio de parada del algoritmo se cumple, ya sea que el error al establecer las correspondencias, con la nueva transformación hallada, se encuentra por debajo de un determinado umbral o bien se cumple con cantidad de iteraciones máxima.

La biblioteca brinda diversos parámetros para cada una de las etapas mencionadas anteriormente que determinan diferentes configuraciones para su utilización.

4.1.1 Modificaciones a libpointmatcher

libpointmatcher es una biblioteca con licencia de código abierto, lo cuál permitió realizar las modificaciones necesarias para el cálculo de la covarianza dentro de la misma biblioteca. Para esto se realizo un *fork* del repositorio público, y se agregaron las funciones necesarias para el cálculo de la covarianza en el caso 2D del minimizador *point-to-plane,* es decir, el cálculo de la matriz de covarianza utilizando las derivadas de la función de error, según se vio en 3.2

4.2 Robot Operating System

`ROS` es un framework para el desarrollo de software para robots que provee una funcionalidad similar a la de un sistema operativo. Cuenta con una colección de herramientas, bibliotecas y convenciones que generan una capa de abstracción tal que permite que la atención del desarrollador esté enfocada en el problema a resolver, abstrayéndolo de los aspectos de bajo nivel.

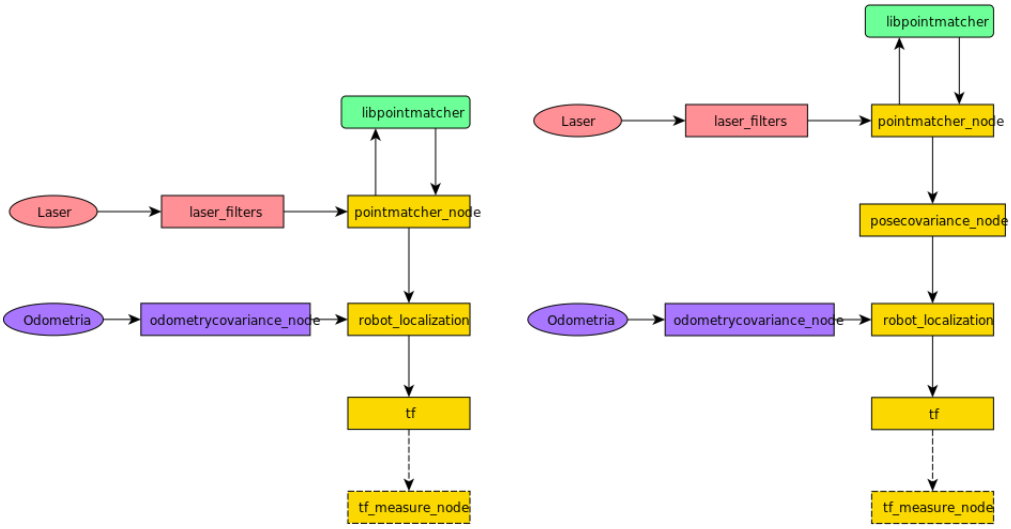
`ROS` provee los servicios estándar de un sistema operativo tales como administración de archivos, abstracción de hardware, control de dispositivos de bajo nivel, implementación de funcionalidades de uso común, pasaje de mensajes entre procesos y mantenimiento de paquetes.

Conceptualmente está basado en una arquitectura de grafos compuesto por *nodos* y *tópicos*. A los primeros se los puede asociar con el concepto de proceso. Los segundos tienen como finalidad establecer un canal de comunicación entre los diversos nodos, de modo que posean un mecanismo de intercambio de información y puedan colaborar entre sí, concretamente recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros.

A continuación se detallan cada uno de estos componentes que constituyen las soluciones propuestas, tanto para el caso en el que se calcula la covarianza según el método propuesto, como para el caso en que la covarianza esta fijada previamente.

4.2.1 Nodo pointmacher_node

Este nodo fue desarrollado para poder integrar la biblioteca *libpointmatcher* al framework ROS. El mismo convierte los tópicos de ROS en estructuras de datos compatibles con la biblioteca, y a su vez, el resultado de los algoritmos de la biblioteca en tópicos de ROS que son publicados para ser utilizados por otros nodos. Además, este nodo es el responsable de controlar el intervalo



(a) Nodos para el cálculo de la covarianza (b) Nodos para utilizar covarianza fija

Figura 4.3: Como se comunican los diferentes componentes o nodos durante la ejecución del sistema de localización

de *keyframes* para optimizar las ejecuciones del algoritmo de ICP. Este nodo recibirá las nubes de puntos del láser, previamente filtradas por *láser_filters* y publicara un tópico con una pose calculada y su correspondiente covarianza.

4.2.2 Nodo láser_filters

Este nodo provisto por ROS, nos permite filtrar la información cruda proveniente de un sensor láser, o sea, la nube de puntos. Podemos filtrar los puntos por diversos criterios configurables, como sea umbrales de distancia mínima y máxima, o ángulo mínimo y máximo.

4.2.3 Nodo posecovariance_node

Este nodo fue desarrollado para simular la utilización de una covarianza fija para ICP, a efectos de comparar este método con el propuesto en el presente trabajo. El nodo recibe un mensaje de pose proveniente del nodo *pointmacher_node* y les asigna una covarianza fija. De esta forma es posible utilizar el mismo esquema de nodos que para el experimento del cálculo de covarianza según el método propuesto, únicamente interponiendo este nodo entre *pointmatcher_node* y *robot_localization*. De esta forma, la covarianza calculada se descarta, y se reemplaza por un valor fijo.

4.2.4 Nodo odometrycovariance_node

Este nodo es similar a *posecovariance_node* pero trabaja sobre mensajes de odometría. Funciona reemplazando la covarianza de dicho mensaje, por valores fijos. Esto es necesario ya que no se dispone de información de covarianza para el sensor de odometría por encoders para los mensajes provistos por los dataset utilizados para los experimentos.

4.2.5 Nodo robot_localization

Este nodo llamado `robot_localization` fue desarrollado bajo el framework ROS y su objetivo es estimar una pose de un robot integrando mediciones de pose provenientes de diversas fuentes. Utiliza un Filtro Extendido de Kalman que modela un cuerpo rígido con 6 grados de libertad (posición y orientación en el espacio 3D) y es capaz de combinar mediciones provenientes de odometría por encoders, una unidad inercial, o cualquier otra fuente que observe posición u orientación como pueden ser un GPS, odometría visual, etc. El nodo posee un modo de funcionamiento 2D, que limita el movimiento a una superficie planar bidimensional. Este modo es el que utilizaremos en el presente este trabajo, ya que nuestros experimentos se basan en un robot con tracción diferencial y recorridos por superficies planas en interiores.

En el método presentado, este nodo se suscribe a dos tópicos diferentes. Uno correspondiente a la odometría por encoders y otro correspondiente al resultado de la registración por ICP. Ambos tópicos publican una estimación de la posición y orientación del robot en el plano, así como la covarianza asociada que modela la incertidumbre de los valores sensados por cada una de las fuentes.

A partir de las estimaciones provenientes de cada una de las fuentes se realizan las correcciones o "updates" del EKF y se obtienen nuevas poses que resultan de la fusión.

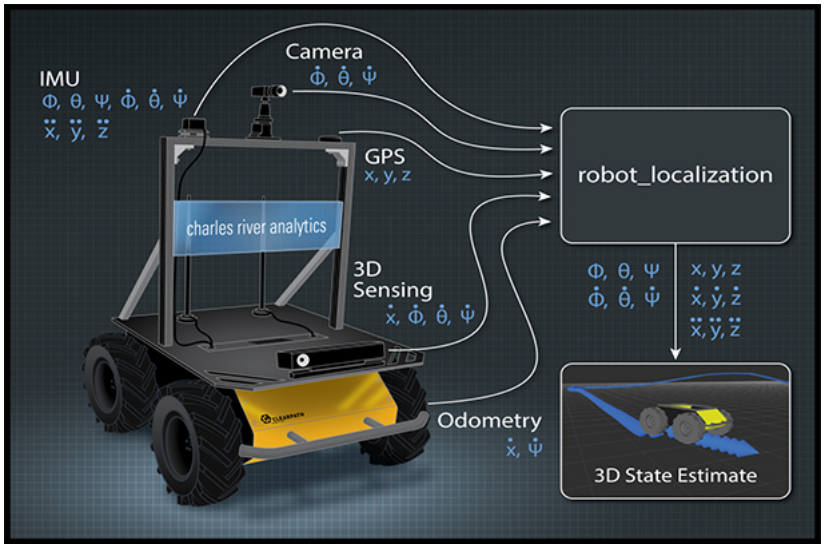


Figura 4.4: robot_localization, diferentes tipos de tópicos

4.2.6 Nodo tf

El nodo tf es central dentro del framework de ROS. Es el que provee y almacena todas las posiciones y orientaciones de cada componente del robot, y todos sus marcos de referencia. Este nodo es el responsable de proporcionar a otros nodos la posición actual, y la relación del robot (representado por el *frame* "base_link") con los elementos que lo componen (en este caso, el sensor láser, representado por el *frame* "láser").

Este nodo provee en todo momento las transformaciones entre cualquier par de *frames* o marcos de referencia. Permite, por ejemplo, trasladar los puntos sensados por el sensor láser, provenientes

del *frame* "láser", a coordenadas del robot, referenciado como "base_link".

4.2.7 Nodo tf_measure_node

Este nodo fue desarrollado como herramienta para poder medir y cuantificar los resultados obtenidos por los experimentos. Su única función es medir las distancias entre los diferentes *frames* y una referencia provista. En este caso, los frames medidos son: "base_link", "base_link_odom" y "base_link_icp". contra la referencia provista por el *ground-truth* del dataset, llamada "base_link_gt".

Las mediciones resultantes se escribirán en un archivo de registro especifico para cada caso.

4.2.8 Nodo pointcloud_node

Este nodo especifico fue desarrollado para acumular las nubes de puntos producidas en cada barrido del láser. Este nodo transforma el resultado del láser a un marco de referencia especifico, utilizando la biblioteca *láser_geometry* provista por ROS. A partir de esta nube acumulada, se puede construir un mapa del entorno según cada método de localización utilizado.

Capítulo 5

Resultados

En este capítulo se presentan experimentos realizados utilizando el método de localización propuesto, y comparándolo con otros métodos que mantienen fija la covarianza. Se describen los elementos empleados para los experimentos (robot, sensores, unidad de procesamiento) y las condiciones en que fueron realizados. El objetivo principal de este capítulo es analizar la precisión en la estimación de la pose de un robot móvil obtenida a partir del desarrollo llevado a cabo en este trabajo.

5.1 Materiales

Los experimentos realizados consisten en la ejecución del sistema de localización sobre un conjunto de datos (o *dataset*) obtenidos a partir de sensores montados en un robot móvil. Se utilizaron dos datasets disponibles públicamente parte del proyecto RAWSEEDS[30] [1].

5.1.1 Unidad de Procesamiento

La ejecución del sistema de localización se realizó sobre una laptop estándar, con la cual se obtuvieron los resultados que se presentan más adelante. Esta computadora posee un procesador Intel Core i5 de 2.4GHz y 4GB de RAM, corriendo un sistema operativo Linux Ubuntu 13.10. Se utilizó la versión Jade Turtle del framework ROS.

5.1.2 Dataset RAWSEEDS

Los conjuntos de datos pre-adquiridos que fueron utilizados para realizar experimentos, pertenecen al proyecto denominado RAWSEEDS. Dicho proyecto tuvo como objetivo la generación de un repositorio de datos obtenidos a partir de diversos sensores montados en un robot móvil, el cual realizó varios recorridos tanto en entornos interiores como exteriores. La particularidad del proyecto es que, además de los datos de sensores, cuenta con información de la pose real del robot en todo momento (*ground-truth*), obtenida a partir del uso de un sistema de localización externo compuesto por cámaras que observan al robot, en el caso del ambiente interior, o de un GPS de alta precisión, en el caso del entorno exterior. De esta forma es posible analizar la calidad de la localización obtenida por los métodos implementados respecto de datos de mayor precisión.

Sobre los conjuntos de datos disponibles, se seleccionaron dos en particular (código **27a** y **27b**) de los generados en un ambiente interior. El plano del entorno (un edificio de la Università

[1] http://www.rawseeds.org/

di Milano-Bicocca, en Milán) y el recorrido realizado por el robot son conocidos de antemano (ver figura 5.1).

5.1.2.1 Robot ROBOCOM

La plataforma utilizada para grabar los datos tiene una locomoción de tipo diferencial (puede girar sobre su eje) y está equipada con unidades de procesamiento para su control y la adquisición de datos. Entre los sensores que posee se incluyen:

- 12 telémetros de ultrasonido, para detectar y evitar colisiones
- 1 Unidad Inercial (IMU) de tres ejes, proveyendo datos de aceleración lineal, velocidades angulares y de orientación respecto del campo magnético terrestre
- 6 cámaras: en configuración monocular, binocular y trinocular
- 2 láser Hokuyo URG-04LX de corto alcance (4 m)
- 1 láser SICK LMS200 de mediano alcance (30 m)
- 1 láser SICK LMS291 de largo alcance (100 m)

5.1.2.2 Sensor láser utilizado

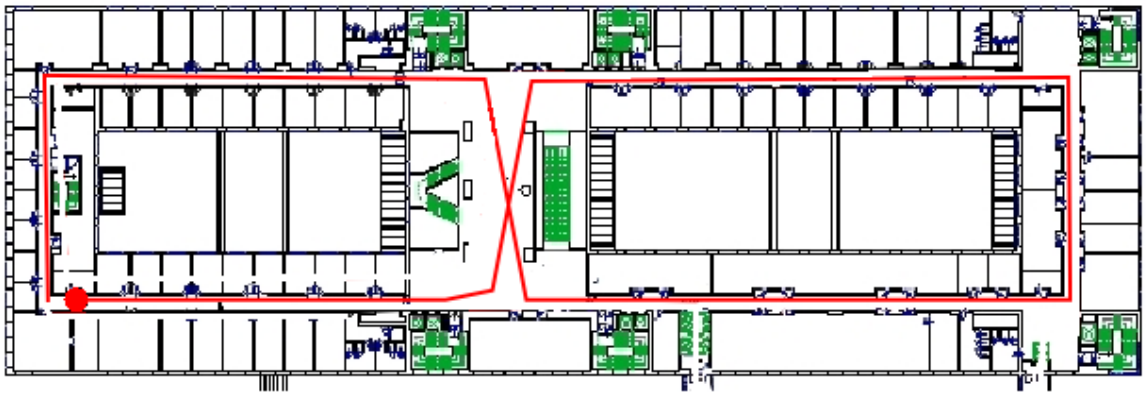
De los sensores láser disponibles en la plataforma se optó por utilizar el sensor de mayor alcance (LMS291) con el objetivo de contar con la mayor información posible ante cada muestreo. Es decir, se buscó evaluar el desempeño del método en condiciones favorables de sensado. Entre las características principales, se puede mencionar:

- Campo de aplicación: ambientes interiores y exteriores.
- Frecuencia de muestreo: 75 Hz.
- Distancia mínima de sensado: 0 m.
- Distancia máxima de sensado: 80 m.
- Campo de visión:180 °.
- Resolución angular: 0,25 °.
- Error aproximado a máxima distancia: 10 mm.

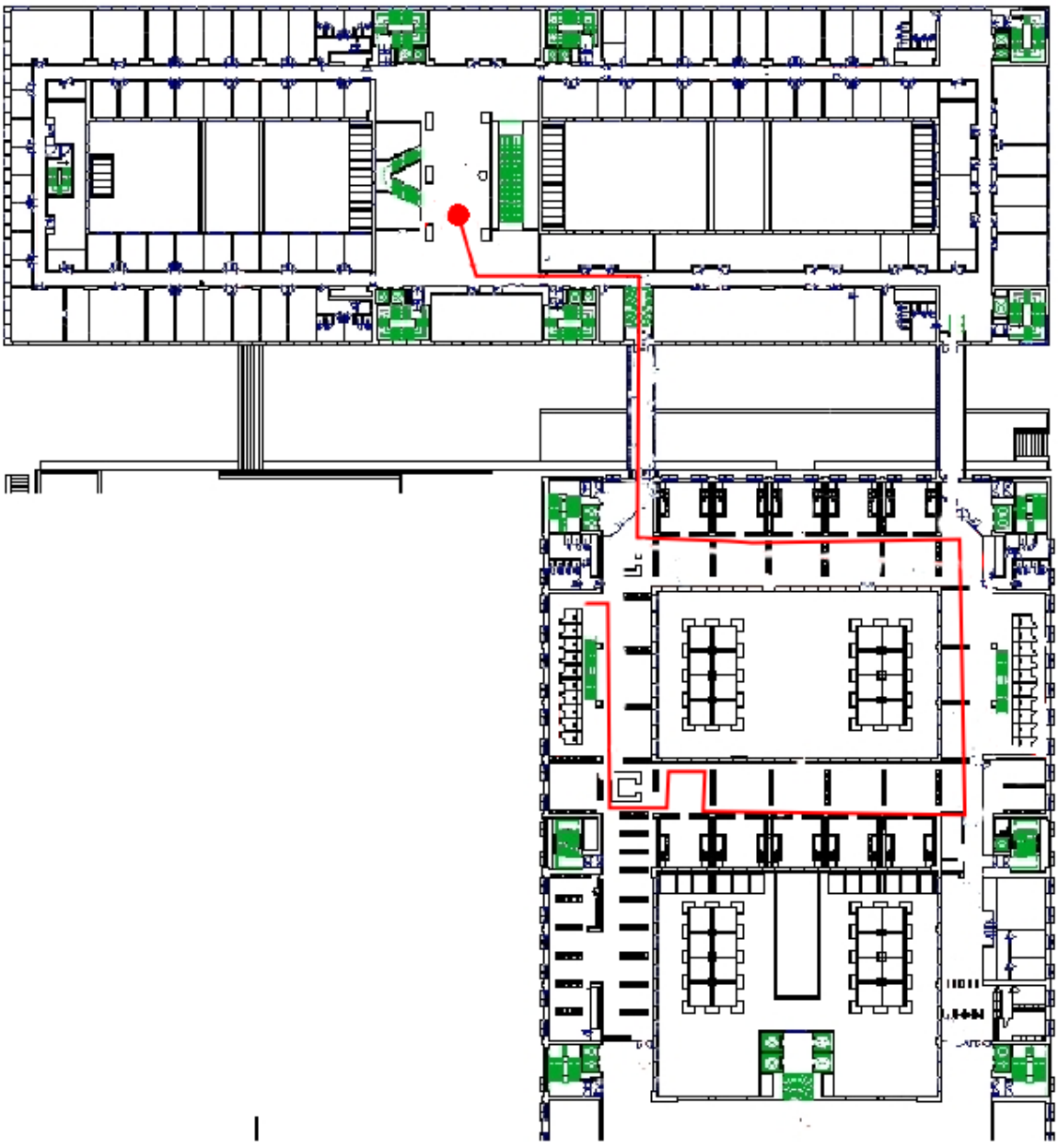
5.2 Experimentos

Se realizaron los siguientes experimentos utilizando los datasets mencionados:

1. Análisis de la covarianza estimada según el método propuesto para el recorrido del dataset, evaluando casos puntuales de interés.
2. Elección experimental de la covarianza para la odometría por encoders según los resultados obtenidos, por no disponer de esa información en el dataset.



(a) Trayectoria 27b



(b) Trayectoria 26b

Figura 5.1: Trayectorias del proyecto RAWSEEDS

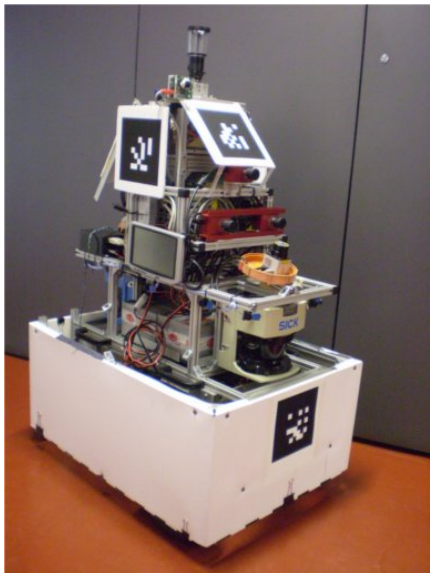


Figura 5.2: Robot ROBOCOM del proyecto RAWSEEDS utilizado para la adquisición de los datos

3. Análisis del error absoluto en la estimación de la posición y orientación, comparándolo con el *ground-truth* y otro método de localización que mantiene fija la covarianza de ICP.
4. Análisis del error relativo en la estimación de la posición y orientación, comparándolo con el *ground-truth* y otro método de localización que mantiene fija la covarianza de ICP.
5. Construcción de un mapa del entorno a partir de las poses estimadas.

5.2.1 Análisis de la covarianza estimada para ICP según el método propuesto

Para poder visualizar los datos de covarianza de ICP, se utilizan elipses que representan un intervalo de confianza del 95 % para una distribución Gausiana bivariada.

A continuación, se analizan secciones de interés particular del recorrido **27b** de RAWSEEDS donde se pretenden encontrar comportamientos esperados de la covarianza. Para analizar los resultados, vemos algunos casos típicos de una registración de dos nubes de puntos mediante ICP. Una curva (5.5), un pasillo con obstáculos (5.4), y un pasillo recto sin información (5.6):

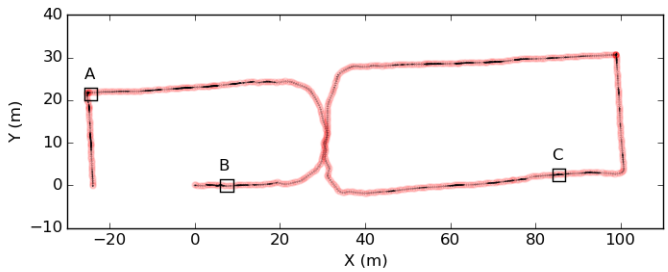


Figura 5.3: Recorrido completo, marcando áreas de interés A, B, C

En el caso del pasillo con obstáculos (ver figura 5.4), se puede observar como la elipse disminuye su tamaño indicando menor covarianza a posteriori, es decir, mayor certeza sobre la información proporcionada por ICP en ese punto. Dado el funcionamiento del algoritmo de registración los obstáculos proporcionan información distinguible para éste, y esto resulta en una transformación mas precisa entre ambas nubes de puntos.

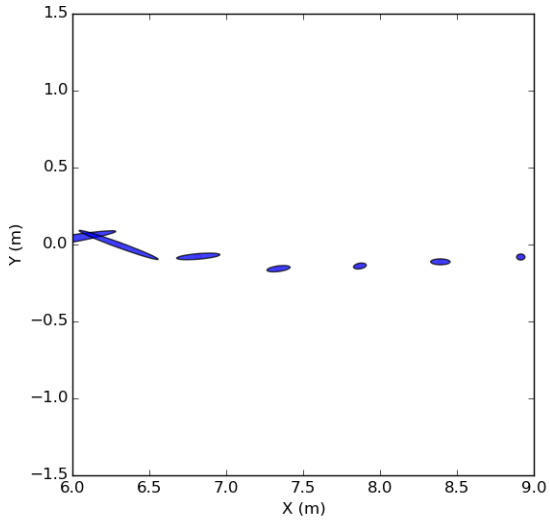


Figura 5.4: Área de interés B

Luego,se puede observar como aumenta de tamaño y orientación a medida que el robot efectúa un giro (ver figura 5.5), y luego vuelve a reducirse cuando el láser recibe nueva información. Esto indica cambios en la certeza de la información de pose obtenida.

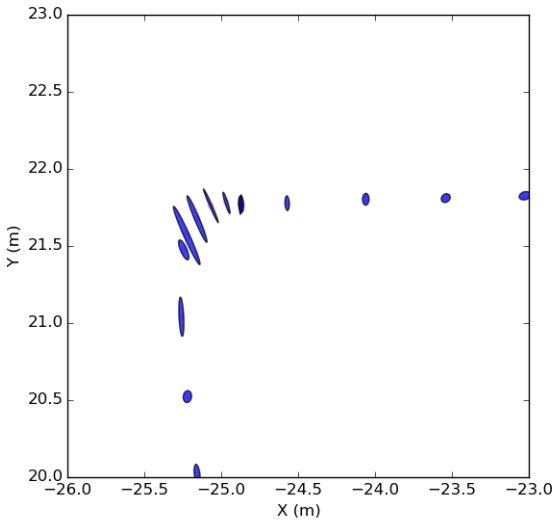


Figura 5.5: Área de interés A

Podemos ver como en un pasillo recto (ver figura 5.6), la elipse resultante es mayor indicando mayor incertidumbre, y su orientación nos muestra que la incertidumbre es mayor en el sentido del pasillo. Este es el resultado esperado, ya que el desplazamiento recto por un pasillo no es distinguible para el algoritmo de registración, puesto que las nubes de puntos provenientes del láser para cada *keyframe* son similares.

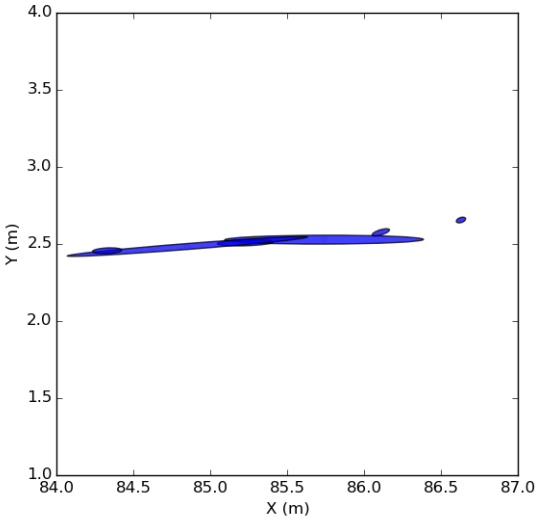


Figura 5.6: Área de interés C

Repetimos el experimento utilizando esta vez el recorrido **26b**, y podemos ver resultados similares en secciones con las mismas características (ver figuras 5.7, 5.8, 5.9, 5.10):

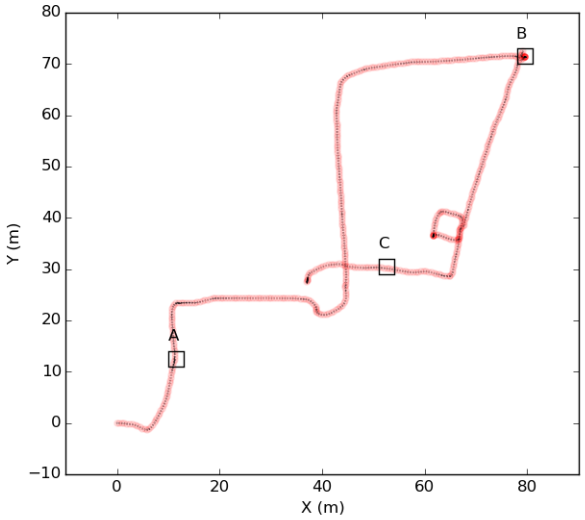


Figura 5.7: Recorrido completo, marcando áreas de interés A, B, C

En la sección marcada A (ver figura 5.8), nuevamente se ve un pasillo en donde al haber pocas diferencias entre cada nube de puntos en la registración, ya que es un pasillo recto sin obstáculos, la incertidumbre del resultado es mayor.

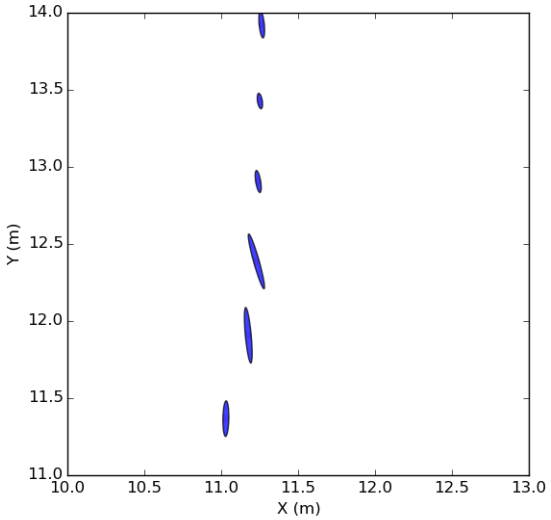


Figura 5.8: Área de interés A

En la sección B (ver figura 5.9), se observa cómo al realizar un giro cerrado de casi 180 grados el algoritmo de ICP no logra una buena registración, y esto se ve reflejado en una mayor covarianza para ese punto. Al continuar el recorrido vemos como la incertidumbre se reduce nuevamente.

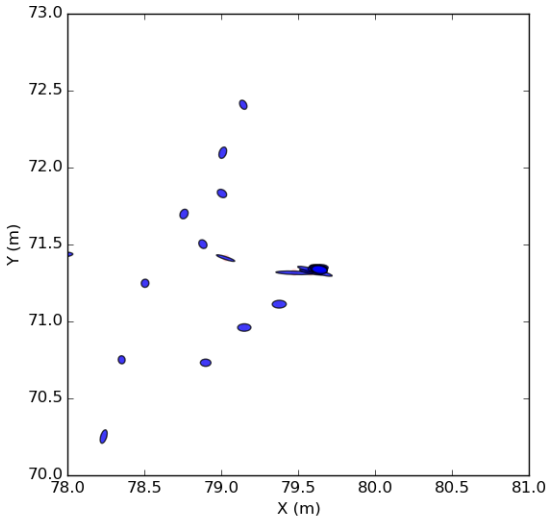


Figura 5.9: Área de interés B

Finalmente, en C vemos un pasillo en donde al haber obstáculos estos proveen más información al algoritmo de registración, y por ende resulta menor la incertidumbre en el resultado.

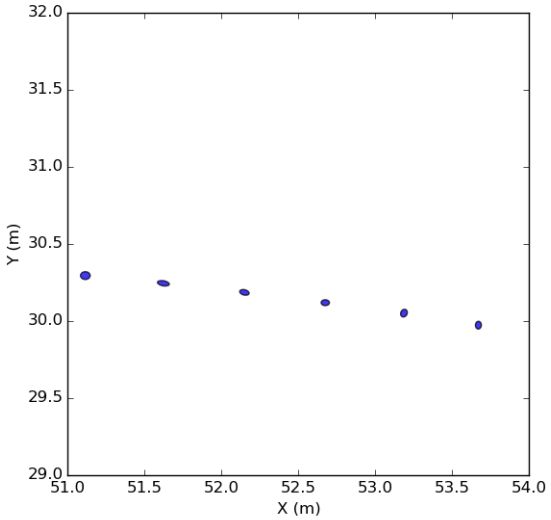


Figura 5.10: Área de interés C

5.2.2 Métricas de error utilizadas

Se proponen dos métricas diferentes para medir el error en la estimación de la pose del robot, el error absoluto y el error relativo.

El error absoluto de posición se mide para un instante de tiempo t como la distancia en el plano entre la pose estimada del robot y la pose registrada en el *ground-truth*. Análogamente, para medir el error absoluto de orientación se calcula la diferencia entre el ángulo de orientación del robot y el ángulo de orientación registrado en el *ground-truth* para ese instante t.

En cambio, para medir el error relativo se toma un intervalo de tiempo $[t_{k-1}; t_k]$ para el cual se analiza el cambio en la posición y la orientación entre esos dos instantes de tiempo. Tomando como referencia el cambio de pose en el *ground-truth* para este intervalo de tiempo, se mide entonces la diferencia con el cambio en la pose estimada. Para este trabajo, se eligen los intervalo de tiempo de acuerdo a los *keyframes* utilizados en la registración por ICP.

5.2.3 Elección experimental de la covarianza fija para el sensor de odometría por encoders

Los valores de covarianza de las mediciones obtenidas por los sensores determinan el impacto que tendrán en la estimación final obtenida por la fusión realizada mediante el algoritmo de EKF. Es decir, la covarianza puede interpretarse como una medida de "confianza" que EKF tiene en cada medición.

Dado que la información de odometría proporcionada como parte del dataset de RAWSEEDS carece de la información de covarianza asociada a las mediciones, debemos asignarle a esta un valor arbitrario. Sin embargo, no es trivial decidir qué valores explícitos de covarianza permite obtener mejores resultados. Para esto se realizaron experimentos con distintos parámetros de covarianza para la odometría por encoders, analizando el efecto sobre la calidad de la estimación.

Para la covarianza del resultado de ICP, utilizamos la matriz calculada por el método propuesto. En comparación, mostramos también el resultado de asignarle a esta covarianza valores fijos.

La matriz de covarianza fija para la odometría tiene la siguiente forma:

$$\Sigma_{odom} = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix}$$

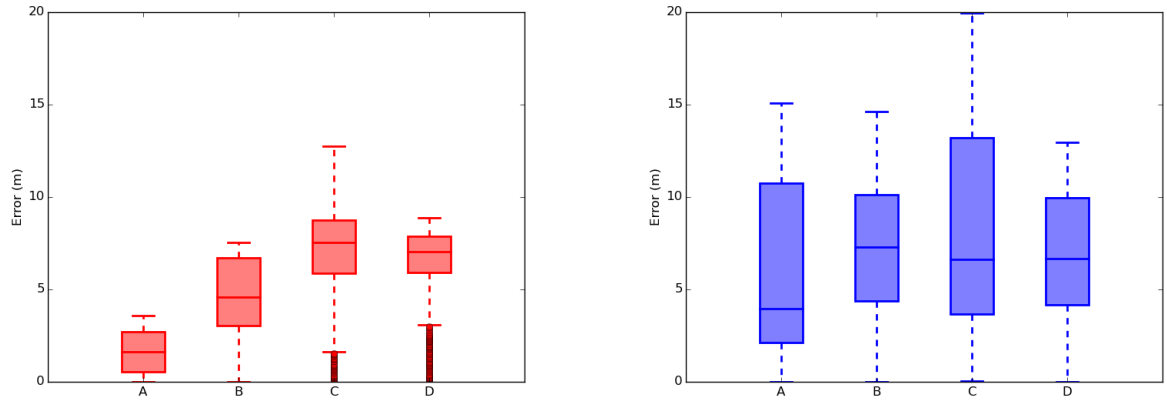
Se analizaran los resultados para los siguientes valores de σ_x^2,σ_y^2 y σ_θ^2:

	A	B	C	D
σ_x^2	10cm	1cm	0,1cm	0,01cm
σ_y^2	10cm	1cm	0,1cm	0,01cm
σ_θ^2	10°	5°	2°	1°

Cuadro 5.1

Se desea evaluar el error cometido por el método propuesto al utilizar tanto una covarianza calculada para ICP, como una covarianza fija. Los resultados son comparados con la información de *ground-truth* provista. En los siguientes gráficos, vemos el error absoluto y relativo medido para cada uno de los experimentos.

En la figura 5.11 se muestran los resultados para el dataset **27b** de RAWSEEDS.

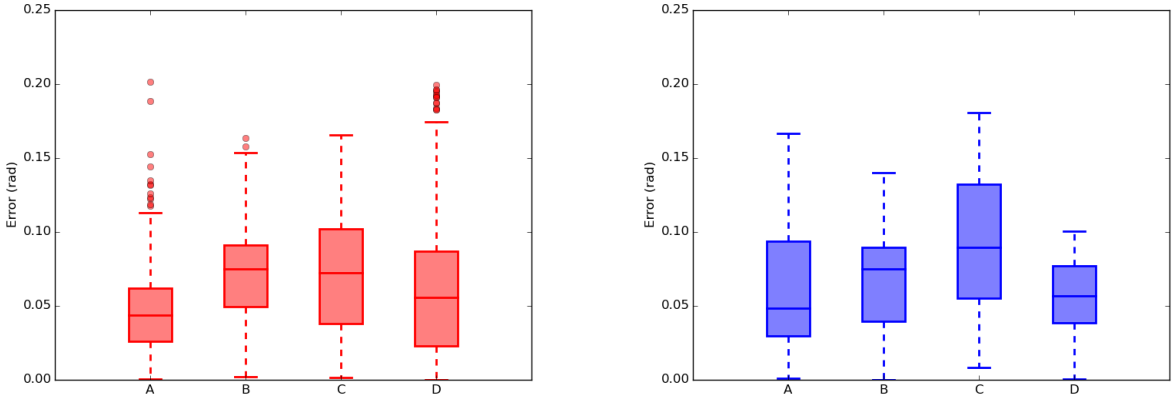


(a) Tukey boxplot del error absoluto de posición utilizando diferentes covarianzas para la odometría, y una covarianza calculada según el método propuesto para la registración mediante ICP

(b) Tukey boxplot del error absoluto de posición utilizando diferentes covarianzas para la odometría, y una covarianza fija para la registración mediante ICP

Figura 5.11: Error absoluto de posición respecto del *ground-truth*

De forma similar, en la figura (5.12) se puede observar el error absoluto en la orientación calculada. Esto es, la diferencia entre el ángulo del robot en el plano XY (5.12a) y el ángulo proporcionado por el *ground-truth* (5.12b) según ambos métodos.



(a) Tukey boxplot del error absoluto de orientación utilizando diferentes covarianzas para la odometría, y una covarianza calculada según el método propuesto para la registración mediante ICP

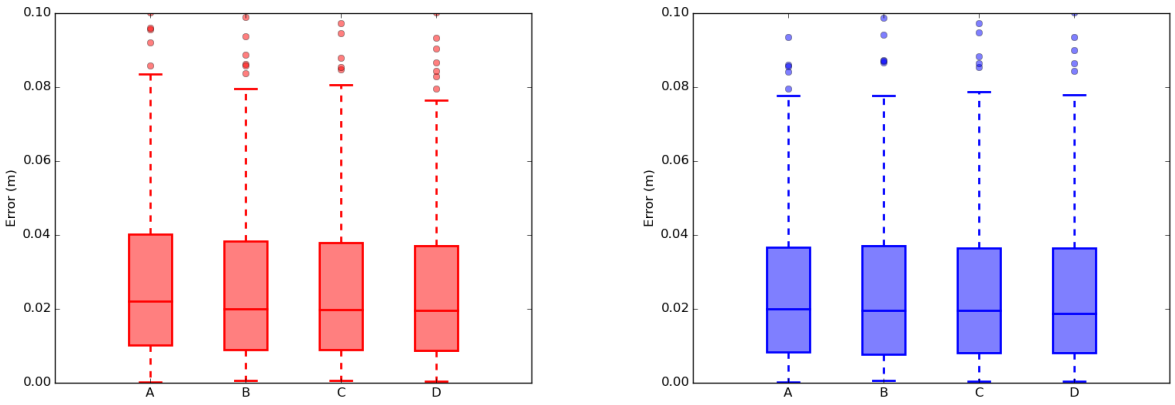
(b) Tukey boxplot del error absoluto de orientación utilizando diferentes covarianzas para la odometría, y una covarianza fija para la registración mediante ICP

Figura 5.12: Error absoluto de orientación respecto del *ground-truth*

Como vemos, tanto en el error absoluto de posición como en el error absoluto de orientación,

los valores promedio del error son menores para el experimento A. Esto se cumple en ambos casos, tanto utilizando una covarianza calculada como una covarianza fija para ICP.

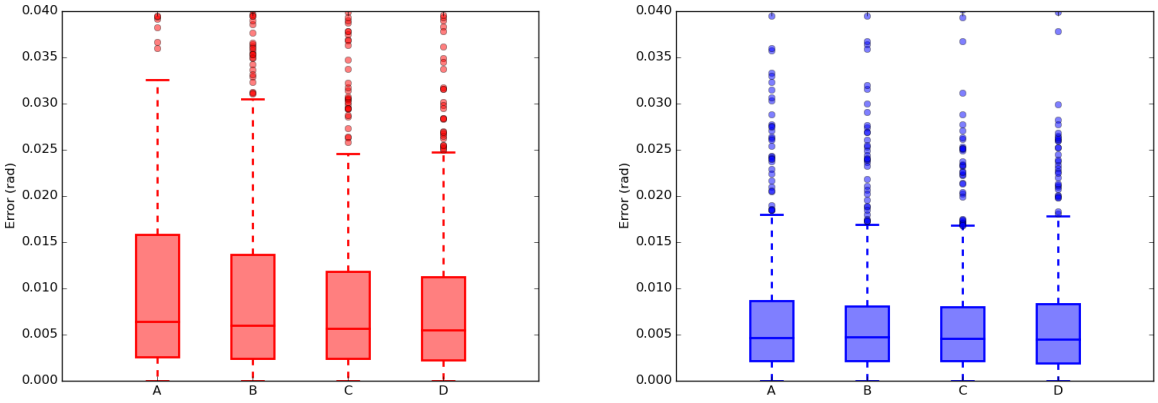
A continuación, se analiza el error relativo para los mismos experimentos. El error relativo es tomando la diferencia entre la nueva traslación y rotación calculadas, y la traslación y rotación proporcionadas en el *ground-truth* en el mismo intervalo de tiempo.



(a) Tukey boxplot del error relativo de posición utilizando diferentes covarianzas para la odometría, y una covarianza calculada según el método propuesto para la registración mediante ICP

(b) Tukey boxplot del error relativo de posición utilizando diferentes covarianzas para la odometría, y una covarianza fija para la registración mediante ICP

Figura 5.13: Error relativo de posición respecto del *ground-truth*



(a) Tukey boxplot del error relativo de orientación utilizando diferentes covarianzas para la odometría, y una covarianza calculada según el método propuesto para la registración mediante ICP

(b) Tukey boxplot del error relativo de orientación utilizando diferentes covarianzas para la odometría, y una covarianza fija para la registración mediante ICP

Figura 5.14: Error relativo de posición respecto del *ground-truth*

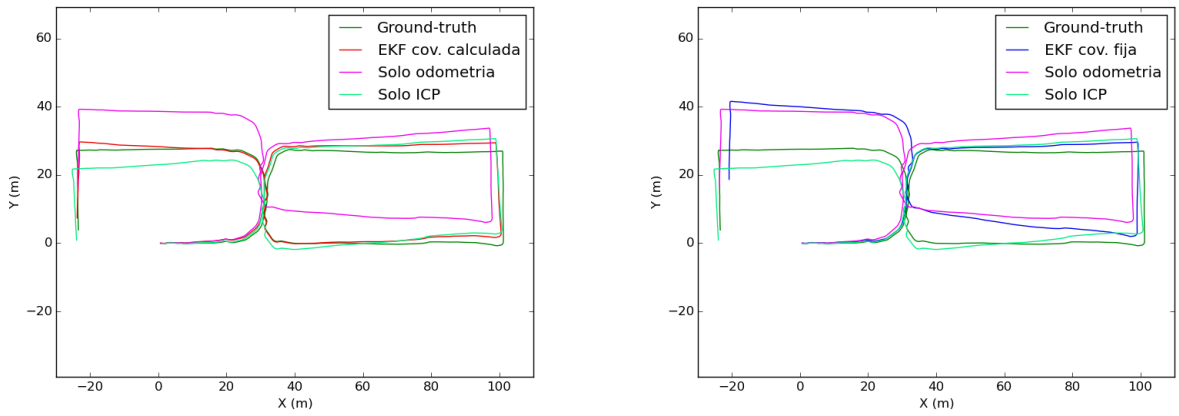
Para ambas medidas de error relativo, el resultado es similar en los cuatro experimentos. Esto indica que en un movimiento a corto plazo, el error cometido no es apreciable, sin importar el método de fusión que utilicemos.

Para el dataset **26b** de RAWSEEDS, se obtuvieron resultados similares, no se muestran ya que no aportan presentan diferencias significativas.

5.2.4 Error de posición y orientación utilizando el método propuesto

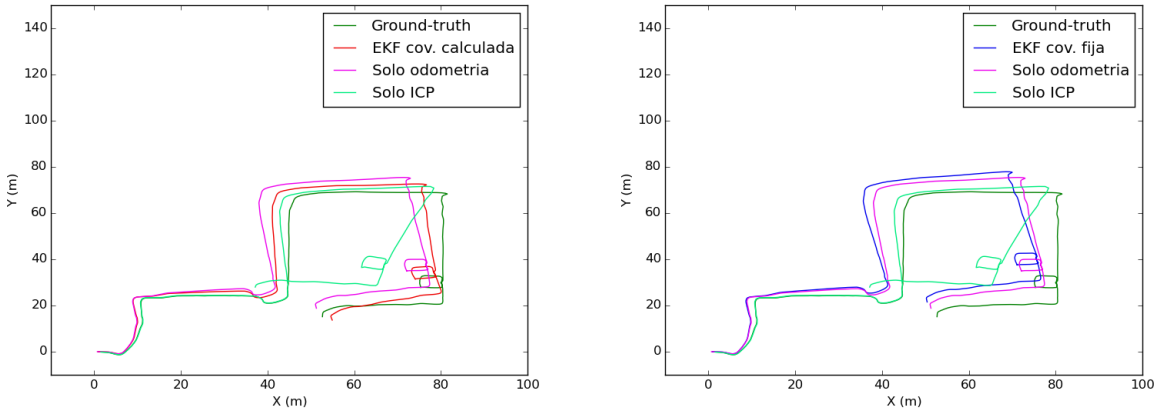
Utilizando la información de *ground-truth* podemos comparar los resultados obtenidos por el método de localización propuesto, y medir el error respecto de este valor de verdad. En este caso se pueden apreciar los recorridos del robot para el método propuesto, entendiéndose por "Sólo Odometría" al resultado de utilizar solo la odometría como fuente de información, "Sólo ICP" el resultado de utilizar solo la información proveniente de la registración de las nubes de puntos del láser, y EKF la fusión de ambas fuentes de información. En el primer caso se presenta la fusión utilizando una covarianza estimada para ICP según el método propuesto, y en el segundo caso vemos el resultado de utilizar una matriz de covarianza fija. Para la covarianza de la odometría por encoders, se utilizo el valor experimental obtenido en el experimento A del apartado anterior (ver cuadro 5.1).

A continuación vemos los resultados para el recorrido **27b** y **26b**



(a) Recorrido utilizando una estimación de la covarianza para ICP (b) Recorrido utilizando una covarianza fija para ICP

Figura 5.15: Recorridos del robot según diferentes métodos para el dataset **27b** de RAWSEEDS

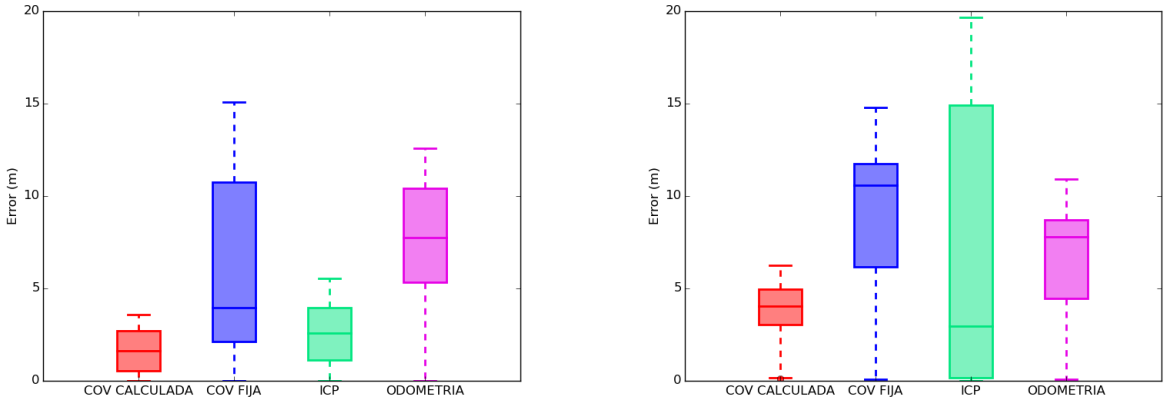


(a) Recorrido utilizando una estimación de la covarianza para ICP

(b) Recorrido utilizando una covarianza fija para ICP

Figura 5.16: Recorridos del robot según diferentes métodos para el dataset **26b** de RAWSEEDS

En los siguientes gráficos, se muestra el error absoluto respecto del *ground-truth* según cada uno de los cuatro métodos mostrados, a saber, covarianza estimada, covarianza fija, sólo ICP, y solo odometría basada en encoders. Nuevamente el gráfico muestra para cada método de localización el error producido respecto del *ground-truth*. Se aprecia cómo el error producido al fusionar la información provista por ICP utilizando la estimación de la covarianza es menor que en los otros métodos evaluados.

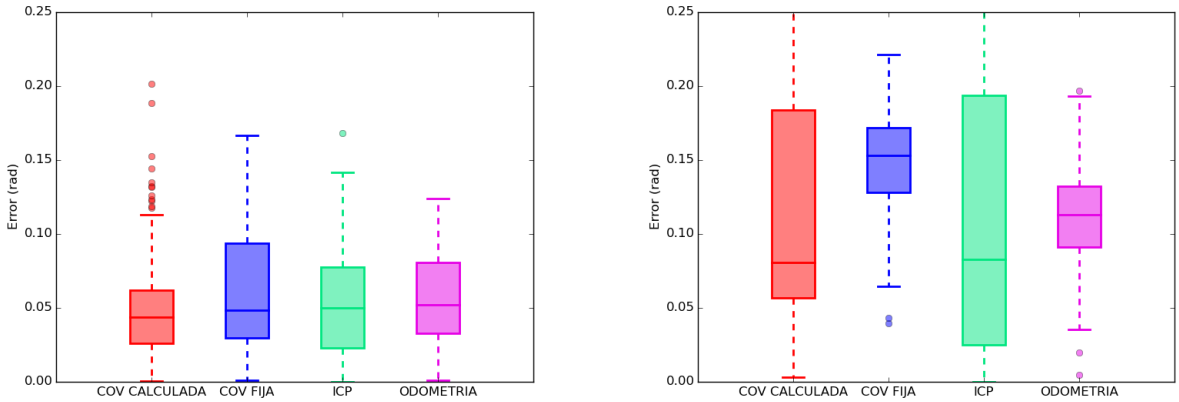


(a) Tukey boxplot del error absoluto de posición para el recorrido 27b

(b) Tukey boxplot del error absoluto de posición para el recorrido 26b

Figura 5.17: Error absoluto de posición

Análogamente, se evalúa el error absoluto de rotación para los diferentes métodos. También se puede apreciar que el error del método propuesto es menor al de los otros métodos evaluados.



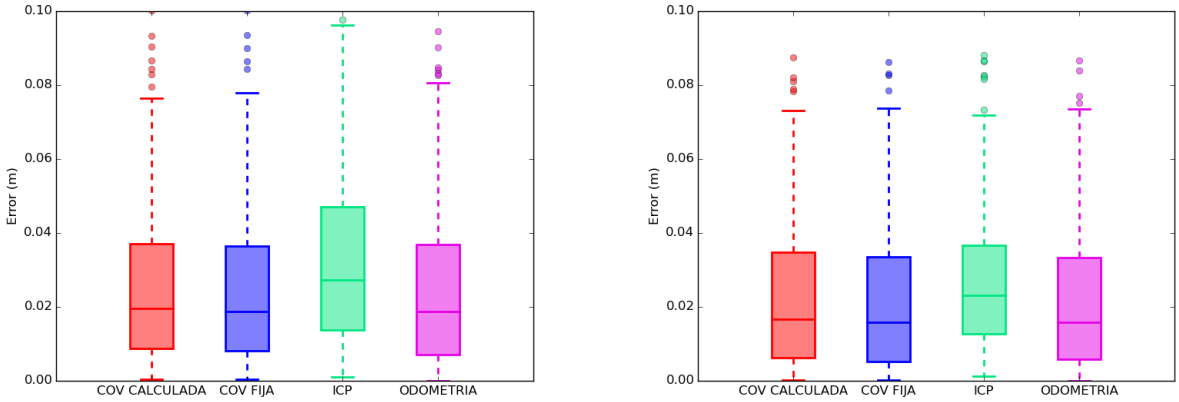
(a) Tukey boxplot del error absoluto de orientación para el recorrido **27b**

(b) Tukey boxplot del error absoluto de orientación para el recorrido **26b**

Figura 5.18: Error absoluto de rotación

A continuación se quiere medir el error relativo a cada nueva registración realizada por ICP. Es decir, para cada nueva transformación resultante del algoritmo, se realiza la fusión con EKF, y se compara esta traslación y rotación con las provistas en el *ground-truth* en el mismo intervalo de tiempo. De la misma forma, en el mismo intervalo se analizan los resultados de la localización por odometría basada en encoders, la localización utilizando solamente ICP, y la localización como el resultado de la fusión por EKF utilizando una covarianza fija. Al igual que en el experimento anterior, para la covarianza de la odometría por encoders se utilizo el valor obtenido en el experimento A (ver cuadro 5.1).

En el siguiente gráfico vemos el error relativo de posición para los métodos mencionados. En este caso, los valores de error calculado resultan similares entre los cuatro métodos.



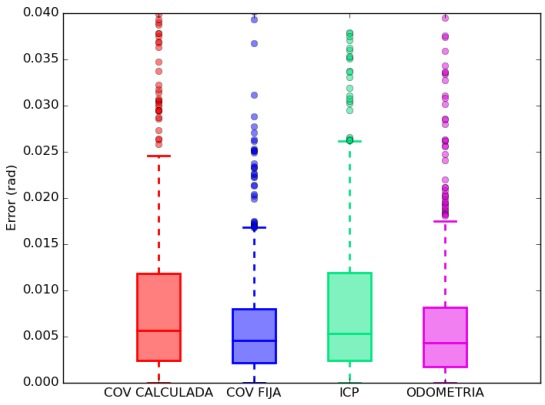
(a) Tukey boxplot del error relativo de posición para el recorrido **27b**

(b) Tukey boxplot del error relativo de posición para el recorrido **26b**

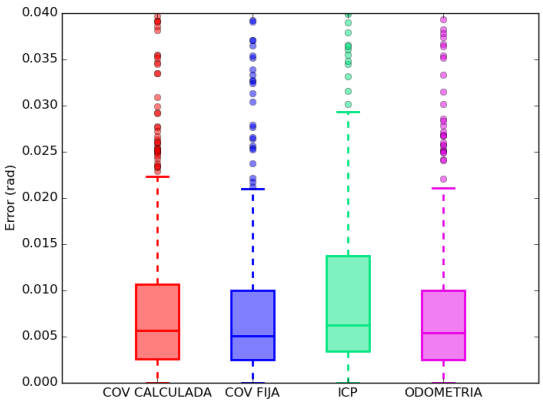
Figura 5.19: Error relativo de posición

De la misma forma que en el error relativo de posición, vemos el error relativo de orientación

para ambos recorridos, obteniendo también un resultado similar para los métodos utilizados para ambos recorridos.



(a) Tukey boxplot del error relativo de orientación para el recorrido **27b**



(b) Tukey boxplot del error relativo de orientación para el recorrido **26b**

Figura 5.20: Error relativo de orientación

5.2.5 Análisis del mapa generado por las nubes de puntos

El método propuesto permite generar una reconstrucción del entorno (un mapa) a partir de acumular periódicamente las nubes de puntos correspondientes a cada sensado. Estos mapas fueron reconstruidos utilizando como punto de origen de la posición y orientación del láser según los diferentes métodos de localización vistos, y utilizando también la información de *ground-truth*. El resultado también puede compararse con el plano del entorno de la figura 5.1.

Puede observarse que para los casos de la localización por EKF utilizando covarianza calculada para ICP, la reconstrucción es fiel a la estructura que se observa en el plano esquemático del edificio.

A continuación veremos los mapas de entorno reconstruidos mediante las nubes de puntos para el recorrido **27b** de RAWSEEDS.

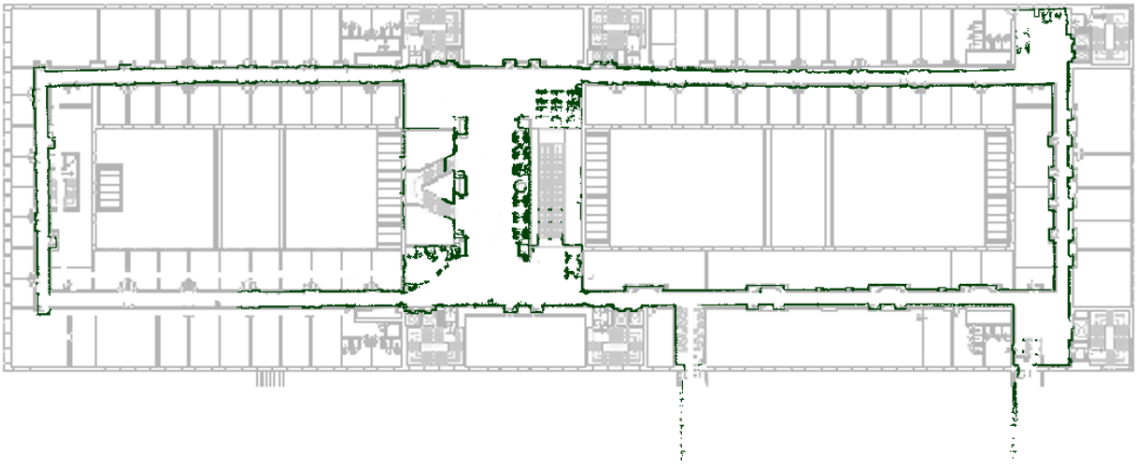


Figura 5.21: Mapa generado según el *ground-truth* para el recorrido **27b**

En la figura 5.21 se ve el mapa generado por el *ground-truth*. Si bien la pose desde la cual se acumulan las nubes de puntos es la proporcionada por el *ground-truth*, vemos imperfecciones generadas por las propias mediciones del láser.

En la figura 5.22 se ve como la mala calidad de la posición calculada por la odometría por encoders produce una deformación considerable del mapa generado. Esto se debe principalmente al error producido por el cálculo de la rotación utilizando odometría.

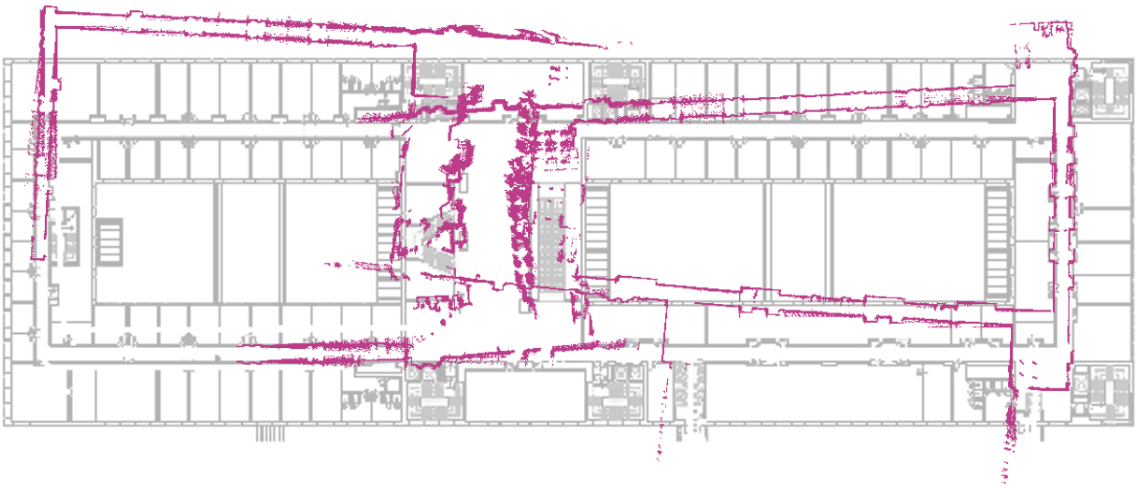


Figura 5.22: Mapa generado por odometría para el recorrido **27b**

Figura 5.23: Mapa generado por ICP para el recorrido **27b**

En la figura 5.23 se ve un mejor resultado al construir el mapa utilizando la pose obtenida por ICP. En este caso, las esquinas del mapa fueron mejor reproducidas, ya que la rotación fue mejor estimada. De todas formas, vemos un desfasaje en el mapa producto de los errores en el cálculo de la traslación del robot.

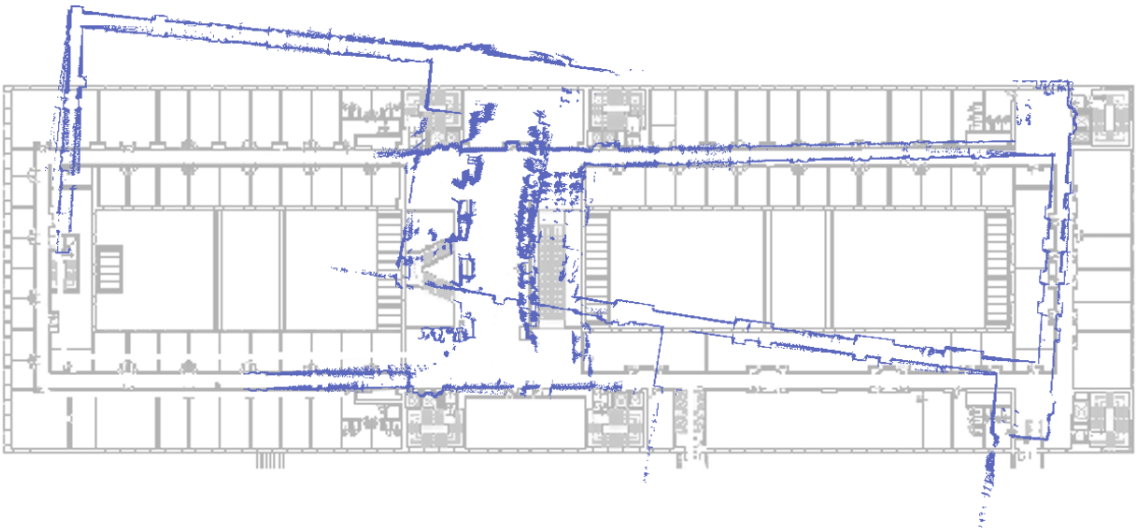


Figura 5.24: Mapa generado por localización por EKF utilizando covarianzas fijas para ICP para el recorrido **27b**

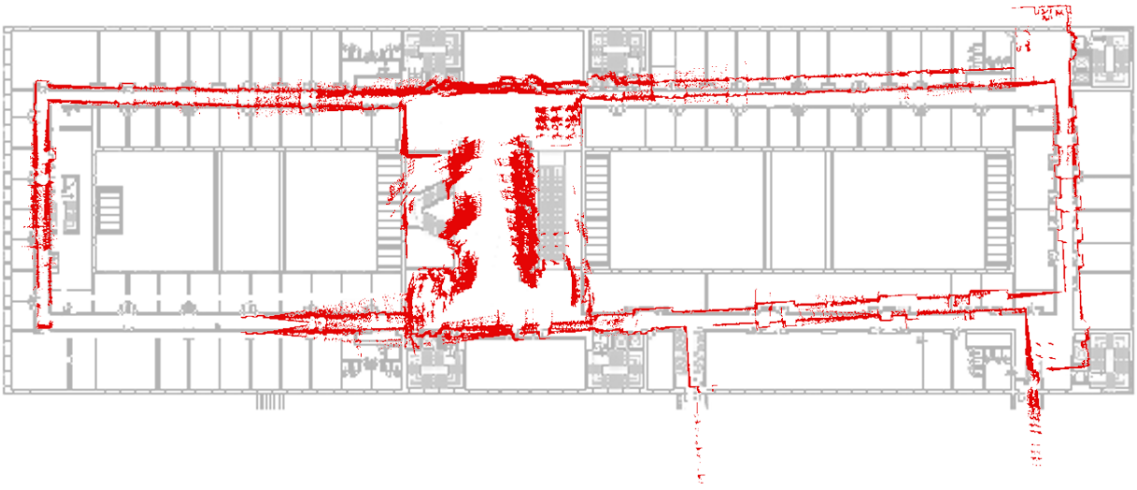


Figura 5.25: Mapa generado por localización por EKF utilizando covarianzas calculadas para ICP para el recorrido **27b**

Se puede apreciar en la figura 5.25 que el resultado de la reconstrucción utilizando la covarianza calculada es más fiel a la original que la reconstrucción utilizando una covarianza fija en 5.24. Mejora también el resultado en comparación a solo utilizar odometría y solo utilizar ICP. Esto se debe a que EKF es capaz de combinar mejor la información, si las incertidumbres modeladas se corresponden con la distribución de los errores en la realidad.

A continuación, mostramos la generación del mapa para el recorrido **26b.** De forma similar al recorrido **27b**, en 5.26 el error en el sensado del láser produce un mapa para el *ground-truth* que es de muy buena calidad pero no es perfecto. Análogamente, en las figuras 5.27, 5.28, 5.29 y 5.30 vemos resultados similares a los producidos por el recorrido **27b**. El mapa generado utilizando la covarianza calculada para ICP en la fusión por EKF es superior a los generados por los otros métodos.

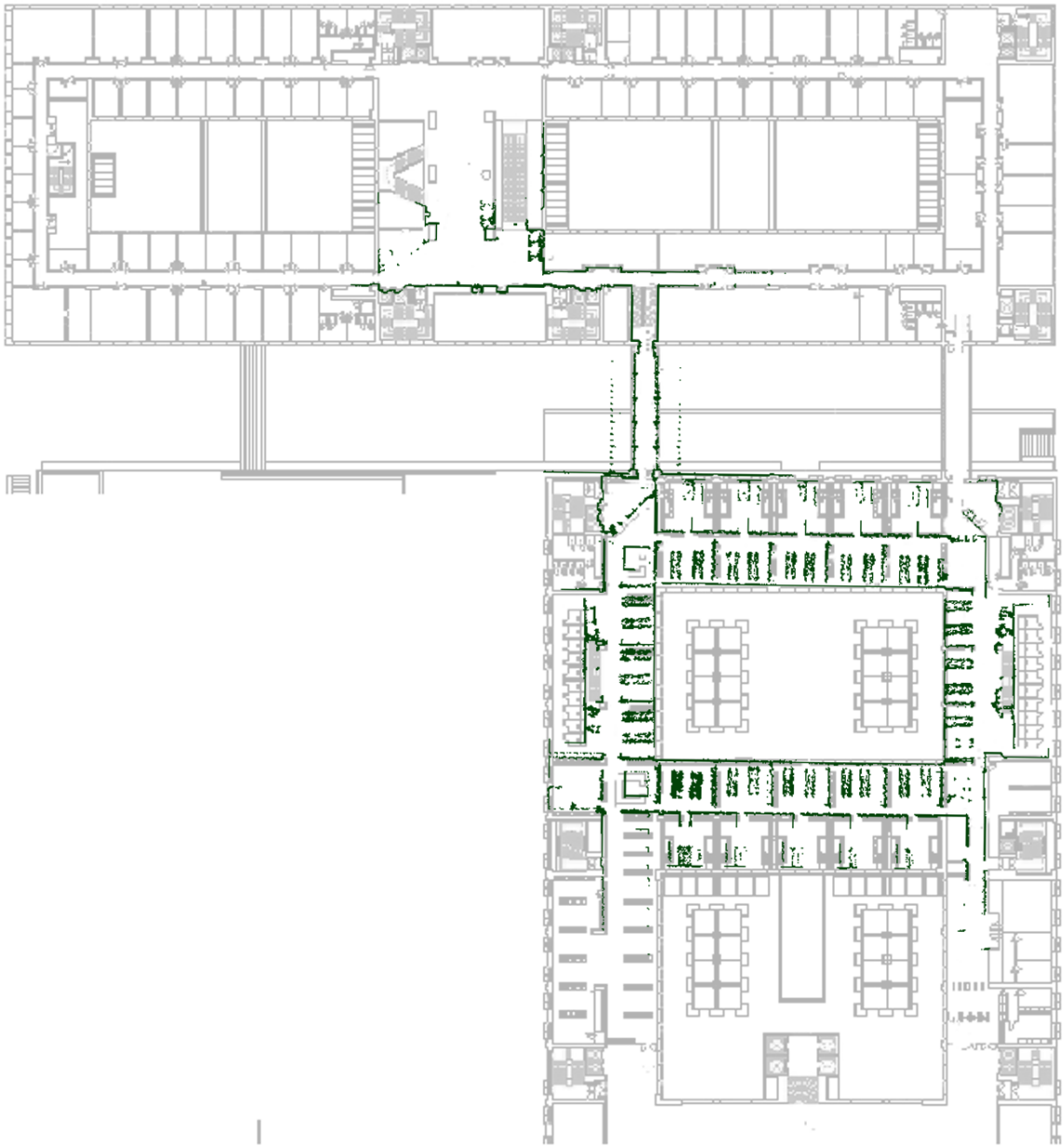


Figura 5.26: Mapa generado según el *ground-truth* para el recorrido **26b**

Figura 5.27: Mapa generado por odometría para el recorrido **26b**

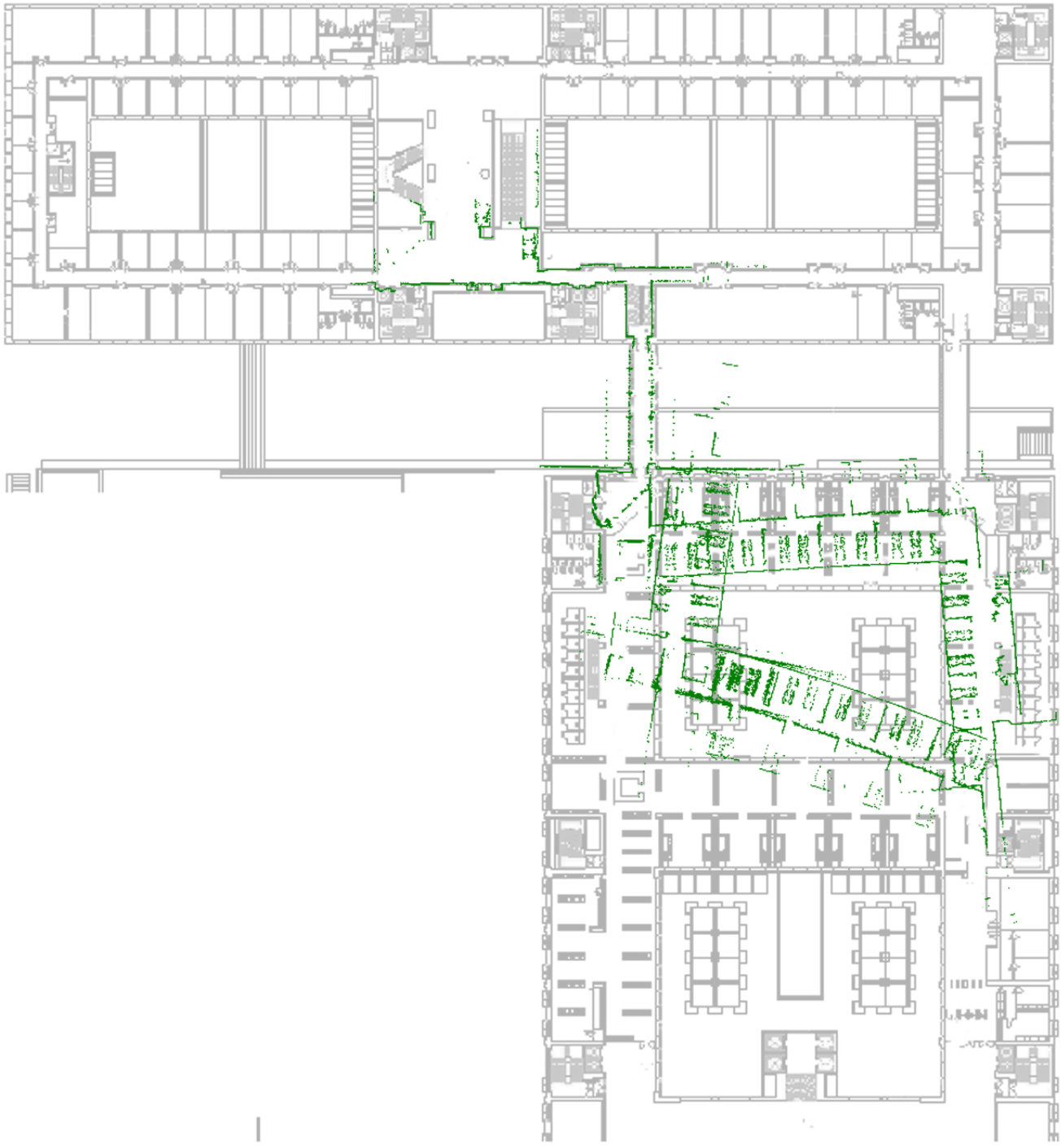


Figura 5.28: Mapa generado por ICP para el recorrido **26b**

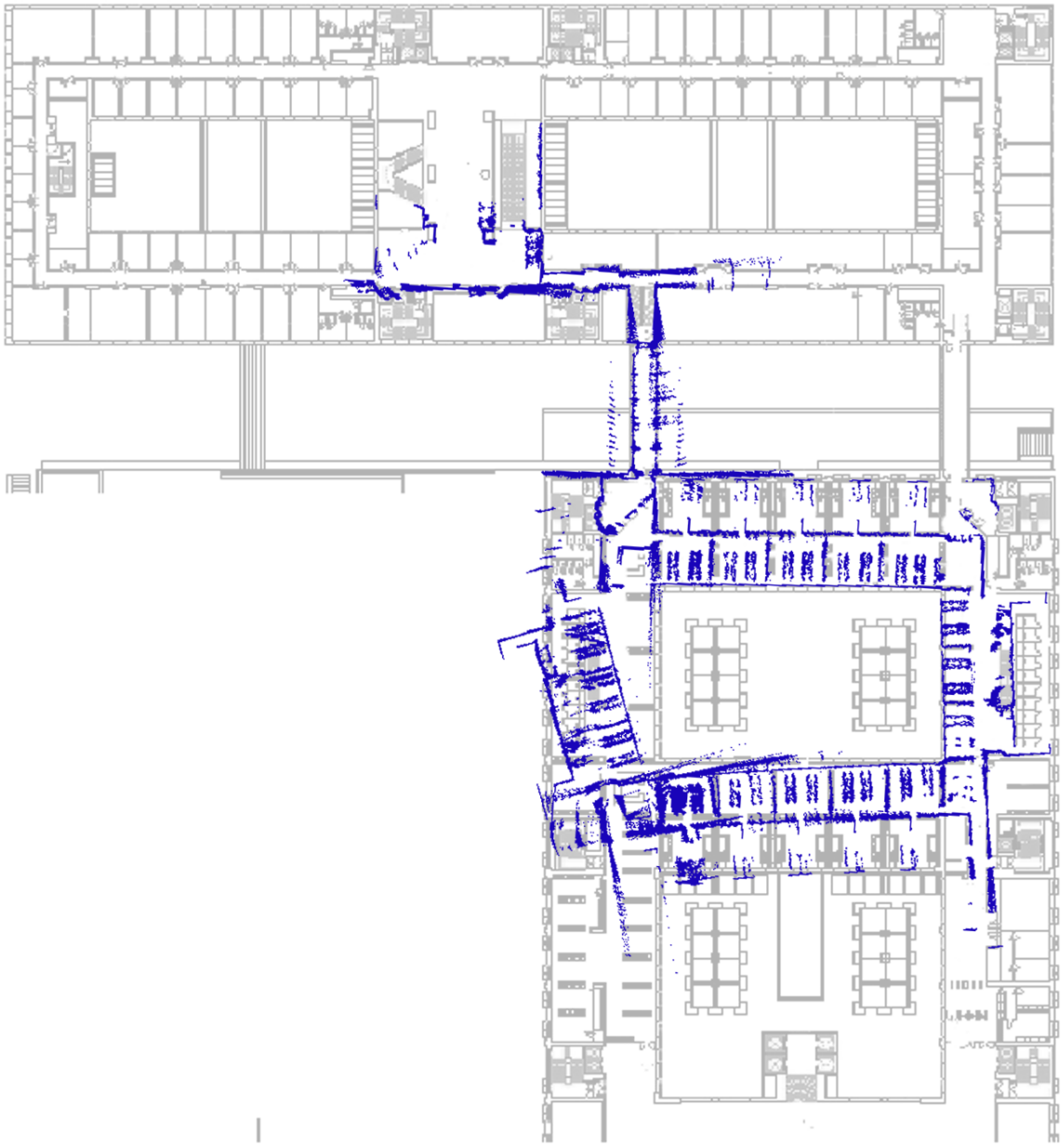


Figura 5.29: Mapa generado por localización por EKF utilizando covarianzas fijas para ICP para el recorrido **26b**

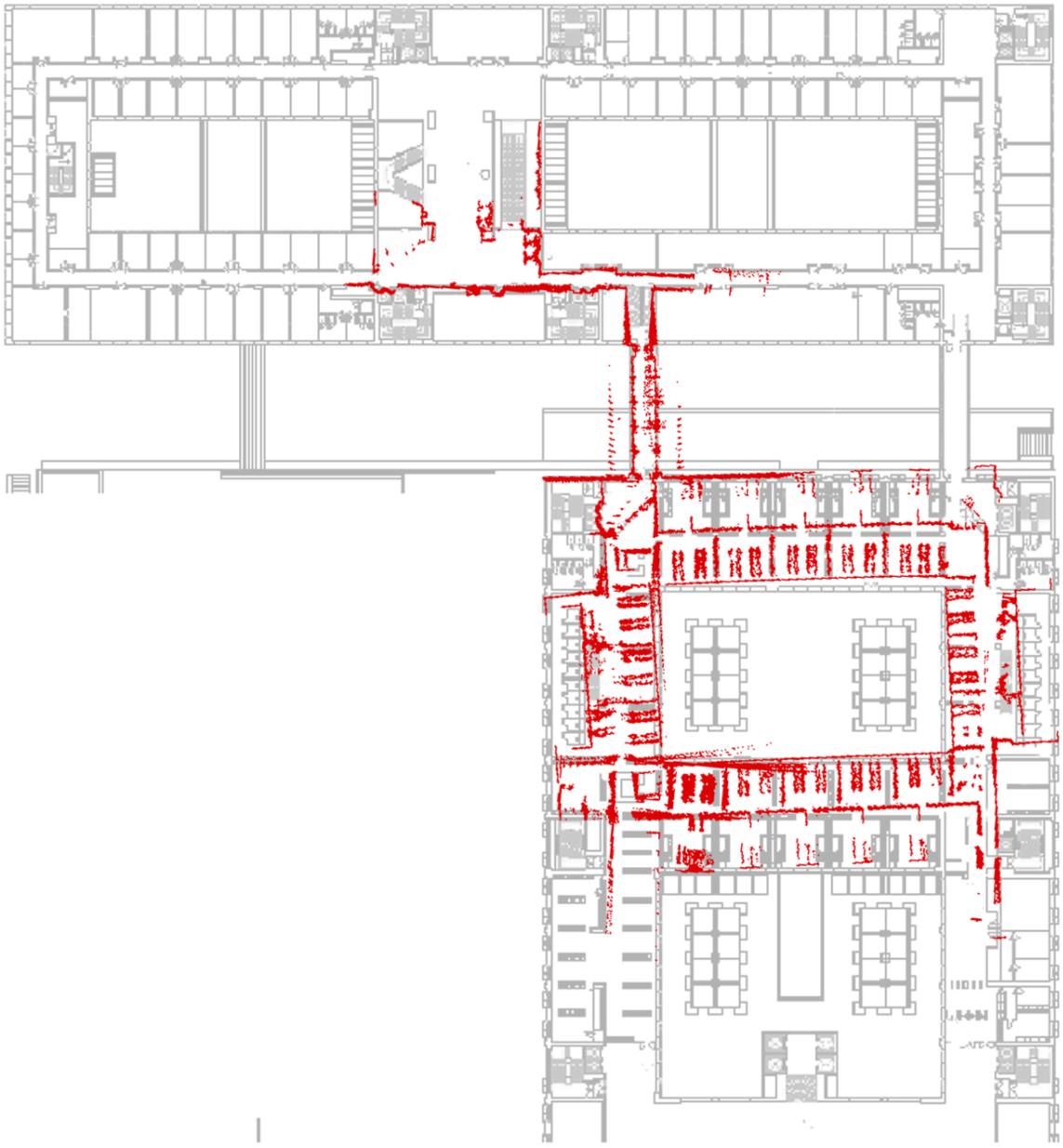


Figura 5.30: Mapa generado por localización por EKF utilizando covarianzas calculadas para ICP para el recorrido **26b**

Capítulo 6

Conclusiones y trabajo futuro

En el presente trabajo se presenta el desarrollo de un método para el cálculo de la covarianza del algoritmo de ICP, su implementación dentro de la biblioteca *libpointmatcher*, y el diseño e implementación de un método de localización basado en el método propuesto, utilizando como fuentes de información la odometría por encoders y las mediciones provenientes de un telémetro láser. Los datos de las distintas fuentes de localización son fusionados mediante un Filtro Extendido de Kalman. Al modelar la covarianza para el resultado de la registración de las nubes de puntos mediante ICP se mejoró el resultado respecto de utilizar una covarianza fija. La mejora en precisión se puede observar analizando las métricas de error absoluto y relativo de posición y orientación respecto de la información de *ground-truth* provista por los datasets utilizados, así como también en los mapas del ambiente construidos a partir de conocer la pose del robot.

Se encontró una matriz de covarianza para modelar el error de la odometría basada en encoders. La misma se obtuvo experimentalmente tras comparar los resultados obtenidos de varias ejecuciones con diferentes parámetros y minimizando el error cometido.

Se analizó la precisión del método de localización propuesto realizando diversos experimentos que consistieron en contrastar los resultados obtenidos contra los valores de *ground-truth* provistos. Para ello se utilizaron dos conjuntos de datos del proyecto RAWSEEDS. Éstos constan de dos recorridos realizados por un robot ROBOCOM equipado con un láser de largo alcance.

Al haber realizado el desarrollo utilizando el framework ROS el sistema resulta fácilmente extensible, permitiendo así utilizarlo como base de futuros trabajos. Entre las mejoras propuestas se encuentran la incorporación de nuevos sensores a través de EKF, como podría ser una unidad inercial (IMU), odometría visual mediante cámaras, o una unidad de GPS para trayectorias en ambientes exteriores. Proponemos también la realización de experimentos sobre un robot que tenga otro sistema de locomoción, como podría ser un robot omnidireccional en lugar de un robot con tracción diferencial como se utilizó en esta tesis.

Bibliografía

[1] Johann Borenstein, Hobart R Everett, Liqiang Feng, and David Wehe. Mobile robot positioning-sensors and techniques. Technical report, DTIC Document, 1997.

[2] Ingemar J Cox. *Blanche: Position estimation for an autonomous robot vehicle, Autonomous Mobile Robots: Control, Planning, and Architecture (Vol. 2)*. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[3] J Borenstein and L Feng. A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots. Technical report, 1994.

[4] J. Borenstein and Liqiang Feng. Measurement and correction of systematic odometry errors in mobile robots. *Robotics and Automation, IEEE Transactions on*, 12(6):869–880, Dec 1996.

[5] Michael Drumheller. Mobile robot localization using sonar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:325–332, 1987.

[6] U. Larsson, J. Forsberg, and A. Wernersson. Mobile robot localization: integrating measurements from a time-of-flight laser. *IEEE Transactions on Industrial Electronics*, 43(3):422–431, Jun 1996.

[7] John J Leonard and Hugh F Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *Robotics and Automation, IEEE Transactions on*, 7(3):376–382, 1991.

[8] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[9] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

[10] Jens-Steffen Gutmann and Christian Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Advanced Mobile Robot, 1996., Proceedings of the First Euromicro Workshop on*, pages 61–67. IEEE, 1996.

[11] Yung Chen and Gérard Medioni. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2724–2729. IEEE, 1991.

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

[13] Zlatko Trajanoski and P Wach. Fuzzy filter for state estimation of a glucoregulatory system. *Computer methods and programs in biomedicine*, 50(3):265–273, 1996.

[14] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[16] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735, 2002.

[17] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[18] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, April 1992.

[19] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 442–448, Oct 2003.

[20] James Manyika and Hugh Durrant-Whyte. *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1995.

[21] W. T. Higgins. A comparison of complementary and kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11(3):321–325, May 1975.

[22] Osder. Navigation, guidance, and control systems for v/stol aircraft. *Sperry Technology, vol. 1, no. 3, 1973, p. 34-41.*, 1973.

[23] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[24] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[25] Andrea Censi. An accurate closed-form estimate of icp's covariance. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3167–3172. IEEE, 2007.

[26] Ola Bengtsson and Albert-Jan Baerveldt. Robot localization based on scan-matching—estimating the covariance matrix for the {IDC} algorithm. *Robotics and Autonomous Systems*, 44(1):29 – 40, 2003. Best Papers of the Eurobot '01 Workshop.

[27] S. Rusinkiewicz. Derivation of point-to-plane minimization & icp stability.

[28] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[29] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 34(3):133–148, February 2013.

[30] Simone Ceriani, Giulio Fontana, Alessandro Giusti, Daniele Marzorati, Matteo Matteucci, Davide Migliore, Davide Rizzi, Domenico G Sorrenti, and Pierluigi Taddei. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371, 2009.