Universidad Nacional Autónoma de México

Facultad de Ingeniería

SLAM Utilizando Modelos Ocultos de Markov

T E S I S

que para obtener el título de:

Doctor

P R E S E N T A :

Oscar Francisco Fuentes Casarrubias

TUTOR

Dr. Jesús Savage Carmona

Ciudad Universitaria, CDMX, 2022

«Once men turned their thinking over to machines in the hope that this would set them free. But that only permitted other men with machines to enslave them.»

Frank Herbert

Agradecimientos

A mi familia, a mi novia.

Este trabajo fue financiado por CONACYT, a través de la beca asignada por el Posgrado en Ingeniería Eléctrica, área de Procesamiento de Señales, UNAM.

También agradezco a DGAPA-UNAM por el financiamiento, a través del proyecto PAPIIT AG101721 “Modelos computacionales para el comportamiento adaptable de robots de servicio y de humanos”.

Índice general

1 Introducción

1.1. Defínción del Problema

El problema de navegar, explorar y elaborar una representación del ambiente no es nuevo para la humanidad. Célebres nombres han alcanzado su lugar en la historia precisamente por sus aportaciones a la cartografía del planeta y sus alrededores. Problemas como obtener la orientación en ausencia del sol, o la más elemental técnica de estimación de distancia (contar pasos) encuentran su símil cuando intentamos que un agente inteligente "explore", "navegue" y "esboce" un mapa.

Si bien "contar pasos" o su equivalente a un robot con ruedas, "la odometría de rueda", realizan estimaciones confiables de desplazamiento en distancias pequeñas, estas estimaciones siempre tendrán un pequeño error, que se acumula con el tiempo, y que hace que las estimaciones dejen de ser confiables, llevando a errores en la representación del mapa, como se observa en la figura 1.1. Donde puede apreciarse un mapa histórico de la península de Baja California, representado como una isla.

Se denomina **SLAM** (por sus siglas en inglés, Simutaneous Localization and Mapping) al problema de Navegación y Representación simultánea del ambiente, ha sido sido abordado de manera histórica por lo cual existe una bibliografía extensa sobre el desarrollo de diversas técnicas para abordar este problema. En el capítulo 2 se mencionan algunos de los algoritmos más importantes y sobretodo aquellos que se encuentran más estrechamente relacionados con el método de **SLAM** propuesto.

Se propone abordar el problema **SLAM** utilizando Modelos Oculto de Markov **HMM**'s (por sus siglas en inglés, Hidden Markov Model), que utiliza varios tipos de sensores, o combinación de sensores, para obtener la representación de un ambiente mientras se navega.

1.2. Justificación

Los robots de servicio como el *Human Support Robot***HSR** [2]1.2 de Toyota o Justina 1.3, del laboratorio de Bio-Robótica de la **UNAM**, están cada vez más cerca de ser una realidad en los hogares; para que estos puedan ser considerados verdaderamente autónomos es indispensable que cuenten con la capacidad de navegar entornos y ambientes donde típicamente coexistimos de forma segura y adaptable. Dichos ambientes están sujetos a cambios y por lo general, no son estructurados como aquellos donde opera un robot industrial. El robot debe contar con la capacidad de adaptación

Figura 1.1: Error de odometría causa un error de estimación en un mapa histórico de la península de Baja California en el año 1650, [1]

al cambio del entorno y capacidad de reacción a obstáculos dinámicos, es decir, no registrados en la representación del entorno.

Figura 1.2: Robot HSR de Toyota utilizado en las pruebas.

DUAL-HMM SLAM, como se nombró al sistema propuesto, es un método que permite, entrenar un agente móvil de manera no supervisada (autónoma), elaborar una representación topográfica del entorno que ha explorado, y de forma simultánea, navegar dicha representación. La representación del entorno en forma de un grafo topométrico permite que el sistema conozca su ubicación, con o sin condiciones iniciales conocidas, como mostramos en nuestro trabajo previo [3]. La propuesta incluye el uso de campos potenciales reactivos para la evasión de obstáculos dinámicos. El uso de memoria para esta representación mediante grafos es mínimo, gracias a esto es

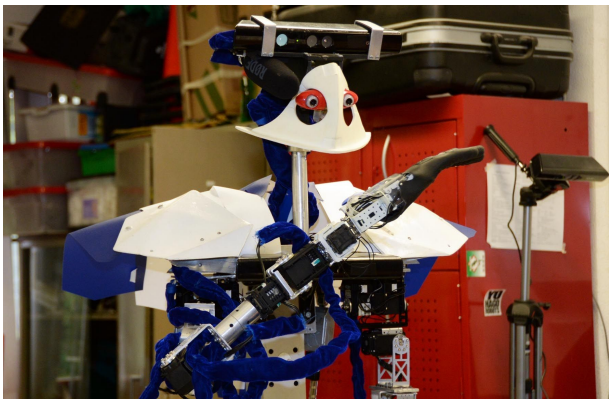


Figura 1.3: Robot Justina del labortorio de Bio-Robótica de la **UNAM**.

posible implementar el método en una gran variedad de robots con menores recursos de memoria y procesamiento (cálculo). Otra característica importante de los modelos obtenidos es la modularidad, ya que permiten agregar y modificar los componentes de los modelos sin necesidad de reentrenar el mapa global, los cuantizadores vectoriales propuestos permiten implementar el método con una variedad de sensores, y/o fusión de sensores. La representación se realiza completamente con los Modelos Ocultos de Markov propuestos, por lo que el uso de mapas es exclusivamente con fines ilustrativos para que el lector cuente con una referencia de lo que sucede; como se mencionó previamente, el modelo no utiliza el mapa sin embargo, el tenerlo disponible puede representar una ventaja.

Se utilizó el simulador Gazebo en ROS (por sus siglas en inglés, Robot Operation System) para probar el método también ha sido probado en competencias internacionales: competencias internacionales **WRS** (por sus siglas en inglés, *World Robot Summit*) 2018 y 2020, y *Robocup* 2021 y 2022. En esta tesis también se reportarán los resultados obtenidos en el laboratorio de Bio-Robótica utilizando el robot real. Para las pruebas se utilizó el paradigma de robot secuestrado [4] (sin condiciones iniciales), representa el peor escenario posible y permite suponer que al tener condiciones iniciales los resultados serán mejores.

1.3. Hipótesis

Los resultados que fueron obtenidos con anterioridad (reportados en [3]) sugieren que podría existir un modelo estocástico capaz de lidiar con la incertidumbre inherente al problema de **SLAM**.

Los Modelos Ocultos de Markov, en conjunto con sus algoritmos asociados, tienen características que permiten una implementación en línea con una gran variedad de sensores para realizar un estimador Bayesiano.

Existen algoritmos de agrupamiento que permiten cuantizar una gran variedad de lecturas para representarlos, de forma discreta y obtener así los corpus de observaciones necesarios para los **HMM's**.

La fusión de sensores detallada en la sección 3.3 plantea un esquema que permitiría obtener más información de los sensores en conjunto que de manera individual.

El conjunto de postulados se propone como una solución a **SLAM** capaz de funcionar de forma robusta, con una variedad de sensores y de forma modular para representar el ambiente explorado por el agente.

1.4. Objetivos

Se propone abordar el problema de la elaboración de una representación del ambiente simultánea a la navegación evadiendo obstáculos de manera autónoma mediante el uso de Modelos Ocultos de Markov. También se proponen métodos de cuantización y fusión de lecturas de sensor provenientes de un Lidar y de una cámara. El ambiente de pruebas se enfoca en el tipo de entorno característico para la operación de un robot de servicio.

Por lo tanto es necesario resolver lo siguiente:

- Navegación Reactiva Autónoma. Que permita al robot explorar, en otras palabras, una conducta segura que permita recolectar datos de posición y de los otros sensores disponibles con la finalidad de obtener un conjunto de datos de entrenamiento.
- Cuantización de Observaciones. Algoritmos capaces de representar lecturas de sensores tipo , Lidar, imágenes de profundidad, e imágenes RGB.
- Fusión de Sensores. Esquemas de cooperación entre sensores.
- Entrenamiento En Línea y Fuera de Línea. Estimación de los **HMM's**.
- Estimación de Odometrías
- Validación Experimental.

1.5. Descripción de la Tesis

La tésis está estructurada de la siguiente forma: En el capítulo 2 se hace una revisión de las herramientas de solución al problema de **SLAM** más utilizadas hoy en día, y se mencionan algunas de sus características más relacionadas con el sistema propuesto. En el capítulo 3 Marco Teórico de las herramientas utilizadas en la solución propuesta de **SLAM**, concretamente Modelos Ocultos de Markov y algoritmos de agrupamiento K-medias y *Affinity Propagation*. En el capítulo 4 se busca conjuntar las ideas de los dos capítulos anteriores en el sistema propuesto como solución de **SLAM**. Finalmente, en los capítulos 5 y 6 experimentos realizados así como las interpretaciones de los mismos y algunas sugerencias de trabajo futuro.

2 Revisión de Literatura

Considerando un robot omnidireccional, con ruedas y varios sensores como, cámaras, Lidars, brújulas, esta tesis propone agrupar las soluciones a **SLAM** en ctres grandes paradigmas (aunque deben existir muchas más): Filtro de Kalman, Filtro de Partículas y **SLAM** por grafos.

2.1. Filtro de Kalman

EKF-SLAM (por sus siglas en inglés,Extended Kalman Filter) la solución a SLAM con mayor aceptación. Basado en la versión extendida del filtro de Kalman propuesto por Smith, Self y Chessman en 1987 [5] [6], consta de tres operaciones básicas.

2.1.1. Movimiento del Robot

El agente cambia de posición de acuerdo a una señal de control por lo tanto, se aumenta la incertidumbre de su ubicación.

$$S_t \longleftarrow f(s_{t-1}, u, n)$$

Donde:

- S_t, estado del robot al tiempo t
- f, modelo movimiento
- u, señal de control
- n, ruido

Descubrimiento

El agente explora y encuentra nuevos puntos de interés, *Landmarks* (por la terminología en inglés), los cuales deberán ser referenciados. Para modelar la incertidumbre de la posición del agente y de los *landmarks* se utiliza un modelo de observación inversa p.ej. ¿Dónde está el *landmark* en el mapa dadas las observaciones registradas por el robot?

$$L_i = g(S_t, \vec{O}_t, y_i)$$

Donde:

- L_i, iésimo *landmark*
- g, modelo de observación
- S_t, estado del robot en el tiempo t
- y_i, lectura del iésimo *landmark*

Redescubrimiento

El agente encuentra *landmarks* en un entorno previamente mapeado, con base en esos hallazgos, determina su posición y la del *landmark*.
El filtro extendido de Kalman cuenta con una etapa específica para cada una de las operaciones (*movimiento, descubrimiento, redescubrimiento*), por lo tanto se puede considerar como un excelente estimador para propagar la incertidumbre de las tres operaciones.

$$y_i = h(S_t, \vec{O}_t, L_i)$$

Donde:

- h, Modelo de Observación Indirecta

La representación se realiza mediante un conjunto de los vectores de cada *landmark* encontrado y el estado del robot.

$$mapa = \begin{bmatrix} S \\ L_1 \\ . \\ . \\ . \\ L_I \end{bmatrix}$$

Donde:

- S, estado del robot

2.2. Filtro de Partículas

La característica principal de este paradigma radica en la construcción de un mapa de ocupación o *grid map*, posteriormente se utiliza el mapa para para identificar su localización en conjunto con otras soluciones estimadas, dicha metodología es similar a la del filtro de Kalman, de ahí toma el termino filtro para ser denominado filtro de partículas. La Localización Adaptable Montecarlo es mejor conocida por su acrónimo **AMCL** (por sus siglas en inglés, Adaptative Monte Carlo Localization), es parte de la familia de algoritmos de Markov. Murphy [7] et. al. propusieron Filtros de Partículas Rao-Blackewllizados como una solución a **SLAM**, la premisa del algoritmo se basa en estimar la probabilidad conjunta a posteriori: $P(\vec{x}_t, m \mid \vec{z}_t, \vec{u}_{t-1})$. Utilizando la siguiente factorización:

$$P(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = P(m \mid x_{1:t}, z_{1:t})P(x_{1:t} \mid z_{1:t}, u_{1:-t-1})$$

Donde:

- x es la posición del robot
- m es el mapa
- Z_t observaciones en el tiempo t
- U_t la señal de control

Para estimar la P aposteriori $P(x_{1:t} \mid z_{1:t}, u_{1:-t-1})$ se utilizan Filtros de Muestreo de Importancia (*Importance Sampling Filters*) como el **SIR** (por sus siglas en inglés, Sequential Importance Resampling), como mencionan Grisetti et. al. en [7]:
“ each particle represents a potential trajectory of the robot. Furthermore, an individual map is associated with each sample. The maps are built from the observations, and the trajectory represented by the corresponding particle”.
[Cada partícula representa una trayectoria potencial del robot. Adicionalmente, un mapa individual se asocia a cada partícula. Los mapas se construyen a partir de observaciones y la trayectoria que representa la partícula correspondiente.]

2.3. SLAM por grafos

Según Grisseti et.al. [8],
“one intuitive way of formulating SLAM is to use a graphe whose nodes correspond to the poses of the robot at different points in time and whose edges represent constraints between the poses”.
[una forma intuitiva de formular SLAM es utilizar un grafo cuyos nodos corresponden a las poses del robot en diversos puntos en el tiempo y cuyas aristas representan las restricciones entre las poses]
Desde un punto de vista probabilístico, **SLAM** puede ser representado como una secuencia de variables aleatorias:

$$P(X_t, M \mid Y_t, U_t, x_0)$$

Donde

- X posición del robot
- M mapa
- Y lecturas de sensor
- U señal
- para cada tiempo t

Para resolver **SLAM** simplemente se busca maximizar la P a posteriori.

Existen diversas y variadas investigaciones para resolver la estimación anterior p.ej. [8] [9].En [10] los autores emplean diversas estrategias de eliminación de variables para reducir el número de dimensiones del problema, dicho enfoque fue utilizado para la elaboración de un *codebook* en el desarrollo de este proyecto. En [11] se propone la solución a un **SLAM** activo "en escenarios donde se tiene información previa del entorno en forma de una grafo topo-métrico".

2.4. ROS SLAM

Como se mencionó previamente, muchas de las soluciones del estado del arte pueden encontrarse en librerías *opensource* como es el caso de **ROS** (por sus siglas en inglés, *Robot Operating System*). En esta sección se mencionarán brevemente los paquetes más utilizados para solucionar **SLAM**.

2.4.1. Sistema Gmapping

Gmapping [12] es un desarrollo con base en filtros Rao-Blackweillizados, fue propuesto por Grisetti et. al. en [13]. Los Filtros de partículas se utilizan para obtener la localización en los mapas de ocupación obtenidos por medio de la técnica de Gmapping, aplicación de Filtros Bayesianos, en los que se usa un conjunto de "partículas" ponderadas por importancia mediante el cual se estima la probabilidad de la posición del agente a posteriori, dadas las observaciones y el mapa. **AMCL**, un conocido miembro de la familia de algoritmos de localización de Markov, es un ejemplo de este paradigma de solución.

2.4.2. Sistema HECTOR SLAM

La diferencia principal entre Hector **SLAM** y otros métodos es que no depende de la odometría de rueda, por lo que este método puede adaptarse fácilmente a otro tipo de agentes móviles, por ejemplo, los drones. En vez obtener la información mediante

la odometría utiliza altas frecuencias de observación en intervalos de distancia/tiempo pequeños, lo que permite utilizar una variante del algoritmo **ICP** (por sus siglas en inglés, *Iterative Closest Point*) para estimar los cambios de posición entre una lectura y otra.

2.4.3. Sistema Cartographer

International's Karto Robotics propuso en [14] un técnica para solucionar **SLAM** mediante grafos, ***Graph-SLAM***, los nodos del grafo representan la posición del robot y las aristas entre ellos corresponden a restricciones entre los nodos, que típicamente están definidos sólo respecto a ellos, con excepción de nodos especiales, donde la matriz de restricciones detecta un *loop-closure*, o cierre de lazo. Los cierres de lazo son indispensables para el desempeño óptimo de *Cartographer* pues corrigen el grafo completo. En la implementación de **ROS** se realiza una comparación e identificación de coincidencias entre las lecturas del sensor, se utiliza el algoritmo de Ajuste de Posición Escazo, **SPA** (por sus siglas en inglés, *Sparse Pose Adjustment*).

La revisión anterior cumple la función de describir brevemente algunos métodos de solución al problema **SLAM**, se destacaron las características más interesantes de cada uno que reslutan relevantes desde la perspectiva del sistema que se propone en esta tesis. La siguiente sección muestra los detalles de cómo se pretende explotar algunas de estas características de cada método en el modelo modelo propuesto desarrollado con base en **HMM's**.

3 Marco Teórico

3.1. Modelos Ocultos de Markov

Un modelo oculto de Markov, **HMM** (por su siglas en inglés, Hidden Markov Model) es un modelo estocástico que relaciona dos variables aleatorias, pero sólo una es directamente observable. Un Modelo Oculto de Markov "λ" está definido completamente por la tupla $[A, B, \vec{\pi}]$, compuesto por las matrices de transición A, de emisión B, y del vector de condiciones iniciales $\vec{\pi}$. Las secciones 3.1.1 y 3.1.2 profundizan en estos conceptos.

En otras palabras, la cadena de Markov no es observable directamente, sólo podemos valernos de un conjunto de observaciones y las probabilidades que tiene cada estado de emitirlas, es decir, se trata de una cadena de Markov parcialmente observable u oculta.

3.1.1. Cadenas de Markov

Una cadena de Markov es un autómata probabilístico, que permite modelar de manera sencilla un proceso (fenómeno) estocástico. Consta de estados que representan el estado interno fenómeno y una matriz de probabilidad de transición entre un estado y otro. La condición necesaria para utilizar estos modelos, y muchos otros basados en programación dinámica (p.ej. los de aprendizaje de refuerzo o *Reinforcemente Learning*), es que los estados cumplan con la propiedad de Markov, es decir, la probabilidad condicional de los estados futuros dependerá sólo del estado actual y es independiente de los estados pasados [15], eq. 3.1.

$$P(X_{n+1} = S_{n+1} \mid X_n = S_n, X_{n-1} = S_{n-1}, ..., X_1 = S_1) = P(X_{n+1} = S_{n+1} \mid \ = S_n) \tag{3.1}$$

$$P = \begin{bmatrix} P_{aa} & P_{ba} \\ P_{ab} & P_{bb} \end{bmatrix} \qquad Pa_{ij} = P(X_t = S_j \mid X_{t-1} = S_i) \tag{3.2}$$

La figura 3.1 ilustra una variable aleatoria discreta que cambia de valor (estado) entre a y b con una probabilidad dada por la matriz de transición (eq. 3.2). Suele ser ilustrada mediante un grafo cuyos nodos representan el estado y los vértices las probabilidades de transición. Son modelos ampliamente estudiados en diversas áreas del conocimiento y con aplicaciones variadas que han llevado al desarrollo de algoritmos propios para el análisis de sistemas modelados con este paradigma, **CTL** ó *Computer Tree Logic*.

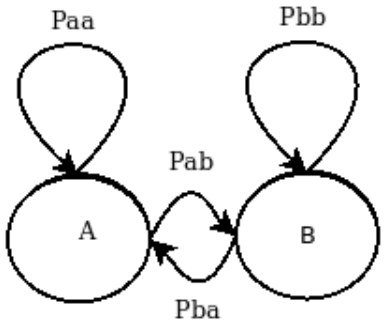


Figura 3.1: Grafo representando una cadena de Markov con las probabilidades de transición de la ecuación 3.2

3.1.2. Modelos Ocultos de Markov HMM's

Como se mencionó en la introducción del capítulo, un Modelo Oculto de Markov es una cadena de Markov, con una matriz de probabilidades de transiciones más una matriz de emisión, con las probabilidad de que dado el estado actual se emita una observación o símbolo. Para el desarrollo de esta tesis se utilizó la variante discreta de dichos modelos, por lo que ambos conjuntos son finitos (estados y de símbolos de observación).

Como se mencionó previamente un **HMM** está definido por dos matrices: La matriz de emisiones B (eq. 3.4) cuyos elementos son, la probabilidad de que dado el estado $x_t = S_i$ se observe el símbolo V_k, $b_j(k) = P(V_k \mid x_t = S_i)$. Finalmente, el vector de condiciones iniciales $\vec{\pi}$ (eq. 3.5) cuyos componentes representan la probabilidad del estado inicial. En caso del paradigma de robot secuestrado, todos los estados iniciales se consideran equiprobables.

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1N} \\ a_{21} & a_{22} & ... & a_{2N} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{N1} & a_{N1} & ... & a_{NN} \end{bmatrix} \tag{3.3}$$

$$B = \begin{bmatrix} b_{11} & b_{12} & ... & b_{1K} \\ b_{21} & b_{22} & ... & b_{2K} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ b_{N1} & b_{N1} & ... & b_{NK} \end{bmatrix} \tag{3.4}$$

$$\vec{\pi} = P(x_0 = S_i) \quad 1 \leq i \leq N \tag{3.5}$$

Los tres problemas de los Modelos Ocultos de Markov

Comúnmente la bibliografía relacionada a los **HMM** hace referencia a los tres problemas de los modelos ocultos de Markov. Evaluación, decodificación y entrenamiento.

Evaluación

Supóngase que se tiene un modelo **HMM** λ y una secuencia de observaciones cuantizadas $\vec{O}$, La evaluación del modelo consiste en encontrar la $P(\vec{O} \mid \lambda)$ o verosimilitud. En teoría esta probabilidad podría ser calculada utilizando argumentos probabilísticos sin embargo, la complejidad computacional crece exponencialmente N^T. El cálculo directo puede apreciarse en la ecuación 3.6.

$$P(\vec{O} \mid \lambda) = \sum_{j=1}^{S} \left(\pi_{s1} \prod_{t=1}^{i=T} a_{s_t.s_{t+1}} b_{s_t - s_{t+1}}(o_i) \right) \tag{3.6}$$

Por lo anterior es inviable el cálculo directo, por lo tanto se utiliza el algoritmo de adelanto *Forward* 3.1.2 para resolver este problema, es decir, se utiliza un enfoque de programación dinámica. A este problema se le conoce comúnmente como la **maldición de la dimensionalidad**. Se declara la variable auxiliar α como la probabilidad de que la observación parcial $\vec{O} = [o_1 = V_k, o_2 = V_k, ..., o_T = V_k]$ provenga de la secuencia de estados $x_1 = S_i, x_2 = S_i, ..., x_T = Si$. Formalmente $\alpha = P(o_1, o_2, ..., o_T, x_1, x_2, x_T \mid \lambda)$. El algoritmo de adelanto permite realizar el cálculo deseado con una complejidad del orden N^2T, en contraste con la complejidad exponencial del cálculo directo. [16].

Algoritmo de Adelanto (o avance) *Forward Algorithm*

- inicio : $\alpha_1(j) = \pi_i$
- t=1,2,...,T: $\alpha_{t+1}(j) = \sum_{i=1}^{N} \left(\alpha_t(j) a_{i,j} \right) b_j(o_{t+1})$
- $P(\vec{O} \mid \lambda) =$ $\sum_{i \in terminal} \alpha_{t+1}(i) = \sum_{i=1}^{N} P(o_1, o_2, ..., o_t, x_t = S_i \mid \lambda)$

Es conveniente definir, de forma análoga, otra variable β que representa la probabilidad de las observaciones parciales pero en sentido inverso con el algoritmo de retroceso o de reversa *Backward*.

Algoritmo de Retroceso o de Reversa *Backward Algorithm*

- inicio: $\beta_T(j) = 1 \quad \forall j$

- t= T,T-1,T-2, ... , 1: $\beta_t(j) = \sum_{i=1}^{N} a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j)$

Al combinar las variables

$$\alpha_t(i)\beta_t(i)$$

se obtiene una expresión sumamente útil:

$$P(\vec{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \tag{3.7}$$

Decodificación

Supóngase ahora que se cuenta con una observación parcial y un modelo λ y se desea encontrar la secuencia de estados que tiene mayor probabilidad de explicar las observaciones, $maxP(\vec{S}_t \mid \vec{O}_t, \lambda)$ de acuerdo con el modelo λ. Gracias al algoritmo de **Viterbi** [17] es posible encuentrar la secuencia de observaciones de máxima verosimilitud o *likelihood* [18].

Algoritmo de Viterbi.

- $inicio:$ $P_1 = \pi_i \cdot b_i(o_1)$

- $iteracin:$ $P_t(j) = \max_{1<i<N} [P_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_t)$

$$S_t(j) = arg \max_{1<i<N} [P_{t-1}(i) \cdot a_{ij}]$$

- $fin:$ $S_t(j) = arg \max_{1<i<N} P_T(i)$

$$P_t(j) = \max_{1<i<N} P_T(i)$$

Entrenamiento

El problema con el mayor nivel de complejidad consiste en estimar los parámetros del modelo λ que mejor se alinean al conjunto de datos de entrenamiento, dichos parámetros corresponden a un conjunto de datos etiquetados, o ejemplos. Se busca generalizar el conjunto de parámetros que no forman parte del conjunto de datos (*dataset*). El método de mayor aceptación para abordar el problema de optimización es el de Máxima Similitud (*Maximum Likelihood*), particularmente el conocido como BAUM-WELCH, aunque el enfoque de gradiente descendente presenta algunas ventajas para la implementación en línea y resolución del problema de **SLAM**; por lo anterior para el desarrollo de esta tesis se eligió utilizar dicho modelo. Ambos algoritmos se detallan a continuación:

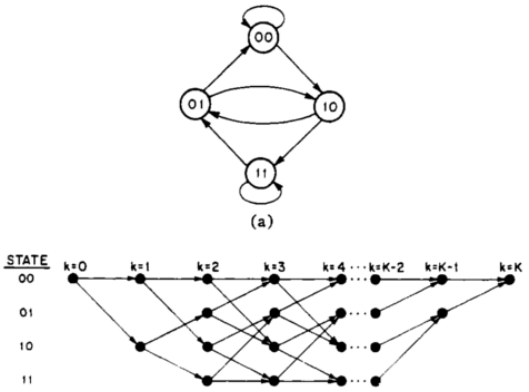


Figura 3.2: Trellis del Algoritmo de Viterbi [19]

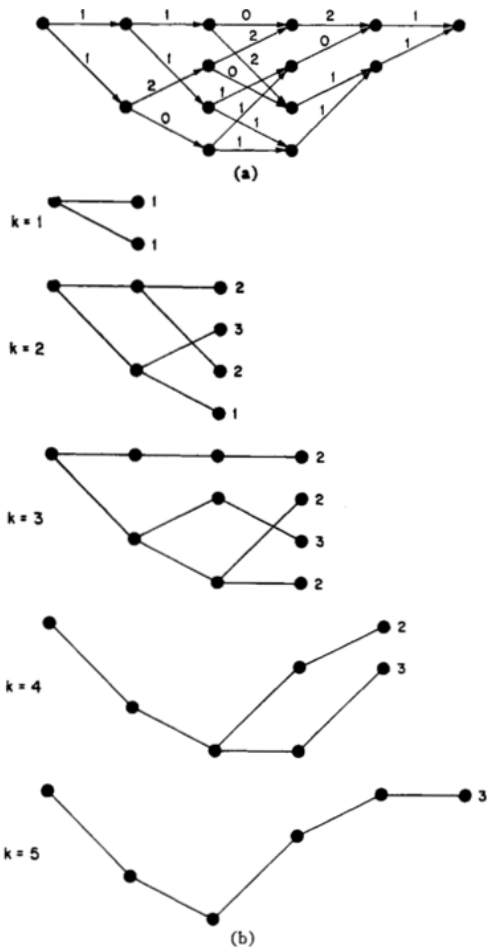


Figura 3.3: Trellis del Algoritmo de Viterbi [19]

Formalmente se busca λ (π ,A,B) tal que maximice $P(\vec{O} \mid \lambda)$, para lograrlo se

propone una nueva variable auxiliar $\gamma_i(t)$, que representa la probabilidad de estar en el estado i al tiempo t dada la secuencia de observaciones $\vec{O}$ y el modelo λ, tomando en cuenta el teorema de Bayes se tiene que:

$$\gamma_i(t) = P(S_t = i \mid \vec{O}, \lambda) = \frac{P(S_t = i\vec{O} \mid \lambda)}{P(\vec{O} \mid \lambda)} \tag{3.8}$$

Usando la ecuación 3.7 para definir la verosimilitud de las observaciones se propone la notación P para expresarlo en las siguientes ecuaciones.

De forma similar, la variable auxiliar $\xi_{ij}(t)$, se forma con la probabilidad de estar en el estado i en el tiempo t y transicionar al estado j en el tiempo $t+1$ dadas las observaciones $\vec{O}$ y el modelo λ. Procedemos de forma similar que en la eq. 3.8

$$\xi_{ij}(t) = P(S_t = i, S_{t+1} = j \mid \vec{O}, \lambda) = \frac{P(S_t = i, S_{t+1} = j, \vec{O} \mid \lambda)}{P(\vec{O} \mid \lambda)} \tag{3.9}$$

El denominador de la eq. 3.9 es igual al deenominador de la eq. 3.8 y puede escribirse con las variables de los algoritmos de adelanto y atraso según 3.7.

Algoritmo Baum-Welch

$$\xi_{ij}(t) = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{3.10}$$

$$\gamma_i(t) = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(j) \beta_t(j) \tag{3.11}$$

$$\bar{\pi_i} = \gamma_i(1) = \frac{1}{P} \alpha_1(i) \beta_1(i) \tag{3.12}$$

$$\bar{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(\vec{O}_t) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \tag{3.13}$$

$$\bar{b_{jk}} = \frac{\sum_{t=1}^{T-1} 1_{o_t = v_k} \gamma_j(t)}{\sum_{t=1}^{T-1} \gamma_j(t)} = \frac{\sum_{t \in O_t = v_k} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T-1} \alpha_t(j) \beta_t(j)} \tag{3.14}$$

donde la expresión $1_{o_t = v_k}$ es un indicador de el número de veces que se ha observado la variable v_k dado el estado j.

De forma equivalente pueden utilizarse las ecuaciones 3.13, 3.14 y 3.12 para obtener los mismos valores a partir de las variables auxiliares de adelanto y atraso α y β, que pudieron ser calculadas antes para elegir el modelo a utilizar.

Es posible estimar los valores del modelo λ mediante la maximización del valor esperado, lo anterior se puede realizar mediante un simple conteo de ocurrencias. El valor esperado de transiciones de un estado destino (el estado destino puede ser el mismo estado), entre el número de veces que se ve el estado como estado de partida. El valor esperado de observaciones del símbolo k mientras se está en el estado j ,entre el número de veces que se ve el estado como estado de partida.

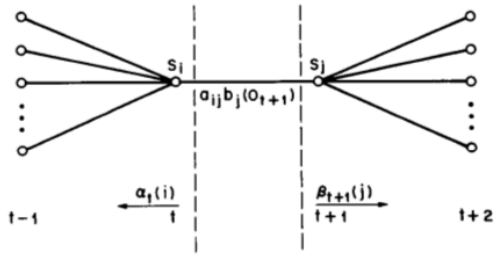


Figura 3.4: Trellis del Algoritmo de BAUM-WELCH [19]

Algoritmo Gradiente Descendente En la implementación del sistema en línea se utilizó una variante que se caracteriza por obtener la estimación mediante un enfoque de descenso por el gradiente, utiliza algunas de las técnicas propuestas por Baldi en [16].

Igual que en todos los algoritmos de aprendizaje, los parámetros del modelo Θ son actualizados y escalados por una taza de aprendizaje η en dirección del gradiente, eq. 3.15.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\partial P(\vec{O} \mid \lambda)^{(t)}}{\partial \theta} \tag{3.15}$$

Por lo tanto se requiere encontrar una expresión del gradiente (o la derivada parcial respecto a cada uno de los parámetros Θ). Para el sistema propuesto en esta investigación $\Theta = \lambda$ y los parámetros son los elementos de las matrices A y B.

$$\frac{\partial P}{\partial a_{ij}} = \sum_{t=1}^{T-1} \alpha_t(i) b_j(O_{t+1}) \beta_{t+1}(j)$$

(3.16)

$$\frac{\partial P}{\partial b_j(O_t)} = \begin{cases} \delta_{j1}\beta_1(1) & t = 1 \\ \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}(O_{t+1}) \beta_t(j) & t \neq 1 \end{cases}$$

(3.17)

Las eqs. 3.1.2 y 3.1.2 muestran que la derivada parcial respecto a cada elemento únicamente modificará esos elementos de la matriz. Posteriormente se realizarán ajustes infinitesimales a cada ejemplo en dirección del gradiente, que depende del estado y de la observación sobre la que se opera.

3.2. Algoritmos de Agrupamiento

El sistema propuesto utiliza un **HMM** discreto, por lo tanto las observaciones y estados deben ser cuantizados, es decir convertir valores continuos a un valores discretos. La técnicas de cuantización representa un tema complejo para el cual existen diversas referencias bibliográficas, es considerada como una de las ramas de la Inteligencia Artificial, particularmente la No Supervisada.

Los algoritmos de agrupamiento se conocen como como "algoritmos de aprendizaje automático no supervisado", a diferencia de los **HMM's** no necesitan un conjunto de datos etiquetado, o *data set* (término utilizado en la literatura de aprendizaje automático). La finalidad de este tipo de algoritmos es etiquetar un conjunto de datos. Formalmente, se representa un vector de orden N como un índice o una etiqueta, o símbolo y los algoritmos de aprendizaje de máquina no reforzado permiten encontrar la equivalencia. En el caso de K medias, es precisamente el índice de cada centroide la etiqueta que representará la lectura, para el caso de *Affinity Propagation* está determinado por el número de ejemplar.

Existen diversos métodos de agrupamiento, se mencionarán brevemente aquellos que resultan de mayr utilidad para nuestro sistema. Se identifica al método **K-medias** como uno de los algoritmos de mayor utilidad. Dicha familia de algoritmos fue investigada y utilizada en los proyectos previos del autor de esta tesis particularmente en [20]. Concretamente la variante propuesta por el Dr. Andrés Buzo et. al LBG [21]. Los esquemas de agrupamiento con mayor relevancia para este proyecto se describen a continuación:

3.2.1. K-Medias

El algoritmo con mayor aceptación y uso es **K-Medias**, separa las lecturas en K cúmulos (o clústeres), el centroide de cada clúster representa a los vectores miembros de dicho clúster. La ventaja principal radica en la capacidad de escalamiento ante grandes números de muestras, la mayor desventaja es que la forma de la medición no respeta su centroide [22].

En este caso podría pensarse que buscamos reducir las dimensiones del *data set* al agrupar un conjunto de lecturas en "K" centroides.

Se detectaron algunos inconvenientes con los algoritmos de agrupamiento (clusterización), ya que la capacidad de utilizar grandes volúmenes de información tiene como detractor que los datos con mayor relevancia (y escasos) del conjunto de entrenamiento o *training set* (es decir aquellas observaciones con mayor relevancia desde el punto de vista de localización y por ende navegación) se ven opacadas por otra información menos útil pero vista en mayor número de repeticiones.

La implementación del sistema fue realizada con la versión por lotes del algoritmo (**Mini batch K means**), que realiza la optimización sobre subconjuntos del data set, es decir, por lotes (o *batches*) lo cual presenta como ventaja principal una convergencia más rápida con poco compromiso en desempeño.

Toda la familia de algoritmos de K-medias pueden ser resueltos una vez más aplicando una maximización de expectativas *Expectation Maximization*. Donde la

expectativa es la pertenencia del elemento x_i al cúmulo cc_j.

$$J = \sum_{i=1}^{S} \sum_{j=1}^{K} w_{ij} \|x_i - cc_j\|^2 \tag{3.18}$$

Donde:

- w_{ij} es una función de pertenencia del elemento x_i al centroide cc_j
- K es el número de centroides
- S es el número total de ejemplos

$$w_i j \begin{cases} 1 \quad si \quad j = argmin(\|x_i - cc_j\|^2) \\ 0 \quad otro \quad caso \end{cases} \tag{3.19}$$

La minimización sucede en dos etapas, primero hacer la derivada parcial de 3.18 respecto a w_{ij} de lo cual se obtiene:

$$\frac{\partial J}{\partial w_{ij}} = \sum_{i=1}^{S} \sum_{j=1}^{K} \|x_i - cc_j\|^2 \tag{3.20}$$

La ecuación 3.20 muestra la etapa de asignación al centroide cuya distancia, norma, sea menor p.ej. se minimiza respecto a w después, de forma análoga, hay que derivar con respecto a cc_j, tal que

$$\frac{\partial J}{\partial cc_j} = 2 \sum_{i=1}^{S} w_{ij}(x_i - cc_j) = 0 \tag{3.21}$$

Lo que lleva a

$$cc_k = \frac{\sum_{i=1}^{S} w_{ij} x_i}{\sum_{i=1}^{S} x_i} \tag{3.22}$$

que puede interpretarse como, recalcular los centroides con las asignaciones del paso anterior.

Es un proceso iterativo que finaliza cuando los elementos no cambian de grupo, o cuando se alcanza un umbral de cambio suficientemente pequeño en los centroides, de forma similar, **mini K medias** procede por lotes. La figura 3.6 muestra que un enfoque por lotes tiene prácticamente el mismo resultados que uno tradicional, con una evidente mejora en el costo de cómputo.

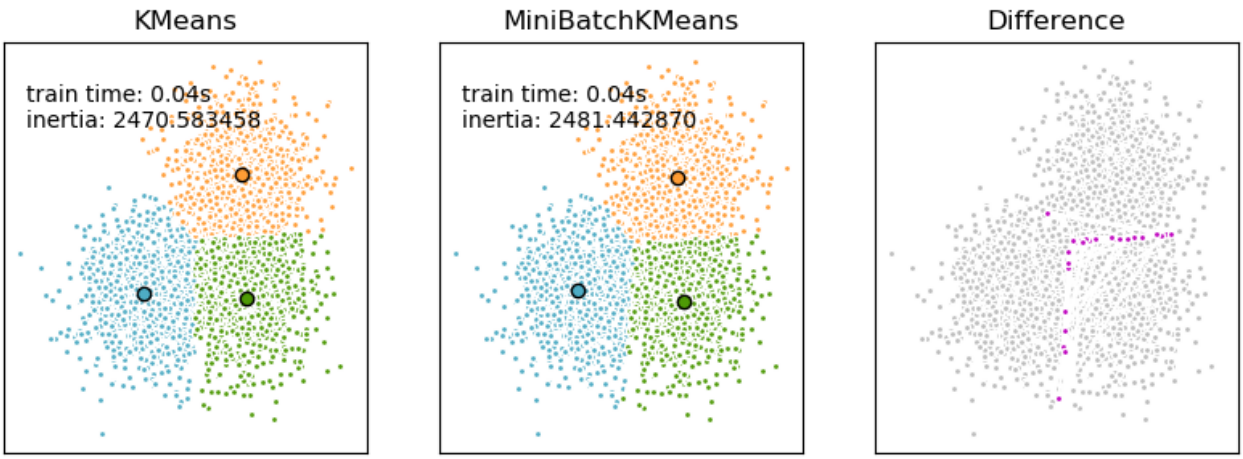


Figura 3.5: Comparación entre los algoritmos K Medias y mini K meadias [22]

3.2.2. Propagación de Afinidad *Affinity Propagation*

Como se mencionó en la sección anterior, una de las ventajas consiste en la minimización del número de parámetros con técnicas de reducción como el análisis de componentes principales, mejor conocido como **PCA** (por sus siglas en inglés, *Principal Component Analysis*). Un algoritmo similar busca propagar la afinidad entre los vectores de características. De forma análoga se propone utilizar la técnica de propagación de afinidad para agrupar el conjunto de ejemplos.

Otra alternativa para crear el Corpus de Símbolos a utilizar por el **HMM** es *Affinity Propagation*, es una alternativa a K-medias, presenta la desventaja de ser caro computacionalmente y por lo tanto preferible en conjuntos de datos (*data sets*) más pequeños. Se logró utilizar con el conjunto de datos de entrenamiento *training set* completo haciendo un muestreo aleatorio de aproximadamente 10,000 ejemplos de entrenamiento. El algoritmo busca representar los datos del conjunto a partir de un subconjunto, elementos considerados "ejemplares", lo cual destaca la ventaja principal por la que decidió utilizarlo. Con base en lo anterior, el símbolo no será una versión deformada, como sucede con los centroides, el algoritmo calcula el número de ejemplares, o símbolos, a partir de los datos que recibe. A diferencia de K medias donde la K puede ser un parámetro importante y difícil de obtener teóricamente. El algoritmo comunica pares de puntos usando dos mensajes, uno de "responsabilidad" r(i,k) 3.24 , que es la evidencia total de que el elemento i es un ejemplar de k y "disponibilidad" a(i,k) 3.23 que es la evidencia de que el elemento i debería elegir k como su ejemplar.

$$a(i,k) \leftarrow min[0, r(k,k) + \sum(r(i',k)] \tag{3.23}$$

$$r(i,k) \leftarrow s(i,k) - max[a(i,k') + s(i,k') \quad \forall k' \neq k] \tag{3.24}$$

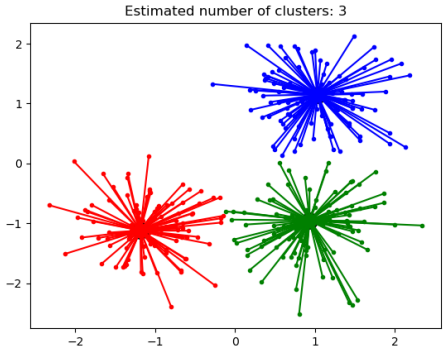


Figura 3.6: Ejemplo de asignación de ejemplares en el algoritmo de agrupamiento Affinity Propagation, Sci-kit learn [22]

3.2.3. Métricas de Desempeño

Los parámetros de métrica que se utilizan en este proyecto sirven para evaluar el desempeño del cuantizador utilizado en el conjunto de datos de entrenamiento. Los parámetros evaluados son los siguientes:

- Inercia *Inertia*. Es la medida que K medias busca minimizar, también conocida como suma de cuadrados intraclúster. Está dada por la ecuación 3.18

- Homogeneidad *Homogenity*. En una escala de 0 a 1, determina si predominan los miembros de una sola clase en cada clúster (cúmulo), valor de 1 significa que el clúster tiene miembros de una sola clase.

- Completitud *Completness*. Es la medida complementaria a la de homogeneidad, indica si predomina la asignación de los miembros de una clase al mismo cúmulo. Una medida de 1 en ambos criterios indicaría una clusterización óptima.

- Validez de Medidas *Validity Measure*. Es la media armónica de la homogeneidad y completitud.

- Ajuste aleatorio de índice **ARI** *Adjusted Rand Index*. Es una medida de similitud registra el consenso entre conjuntos y sus asignaciones, es decir, el consenso entre las etiquetas del conjunto de entrenamiento y las estimadas por el algoritmo (no son de utilidad destacable ya que dependen de la existencia de etiquetas en el *training set*).

- Silueta *Silhouette*. Es una medida que evalúa la definición de los cúmulos del *training set*. Donde a es la distancia media entre una muestra y todos los otros puntos en el clúster y b la distancia media entre una muestra y todos los otros puntos en el clúster vecino más cercano.

3.3. Fusión de Sensores.

La fusión de sensores es una técnica mediante la cual se agrupan (fusionan) sensores distintos, la finalidad consiste en obtener lecturas de parámetros distintos, es común encontrar la aplicación de dicha característica en sistemas como el **HSR** o algún vehículo autónomo. Al día de hoy es un campo fértil de investigación y desarrollo, gracias en gran medida a la Fuera Aérea de EEUU, principal proponente de este esquema [23],incluso en el F-35 uno de los aviones de combate más recientes lo mencionan como uno de sus diferenciadores principales [24]. La técnica ha encontrado un nicho en fechas más recientes en los vehículos autónomos, casas inteligentes, también se ha incrementado el uso generalizado en los sistemas embebidos expertos. La premisa radica en la generación de un modelo de observación del ambiente compuesto por la yuxtaposición de fuentes de información que alimenten al sistema, preferentemente de naturaleza diferente. Existen algoritmos y métodos diversos para generar esta fusión, la dicotomía principal es la de C^3I (por sus siglas en inglés, *Command Control Communications Inteligence*) y la fusión de sensores en tiempo real (típicamente encontrada en sistemas inteligentes embebidos). La primera categoría se concentra en sensores de mediano y alto nivel, mientras que la segunda se ocupa principalmente de las interacciones de bajo nivel propias de un circuito impreso. Algunas de las variantes más comunes (según el nivel) de acuerdo con la Sociedad Internacional de Fusión de Información **ISIF**:

- Fusión bajo nivel. Datos crudos, se unen las lecturas de diferentes sensores para obtener una lectura informativa.
- Fusión nivel intermedio. O de características (*features*) o landmarks. Se fusionan niveles de abstracción de características como esquinas, bordes, etc.
- Fusión alto nivel. Se combinan conceptos simbólicos propios de uno o más sistemas expertos que operan en conjunto, también se le conoce como "fusión a nivel de decisión", p.ej. control difuso.

Una tabla traducida similar a la propuesta en [25] puede ser vista en 3.1

	Fusión *Onboard*	Fusión C^3I
Escala de tiempo	milisegundos	Segundos o minutos
Tipo de Datos	lecturas de sensores	Representación Simbólica
Interacción Humano Máquina	Opcional	Frecuente
Tamaño Datasets	mediano	grande a muy grande
Nivel de abstracción	bajo	alto

Tabla 3.1: Tipos de fusión de sensores.

4 Sistema Propuesto

4.1. Diagrama General del sistema.

Se propone un sistema llamado **HMM-Dual** que fue desarrollado con base en **HMM**. En él se utilizan dos **HMM** en paralelo, cada uno utiliza un alfabeto propio. La intención es establecer una relación cooperativa entre ambos, es decir, cuando un alfabeto no tenga suficientes descriptores la contra parte puede obtener información adicional a partir de la misma lectura del sensor.

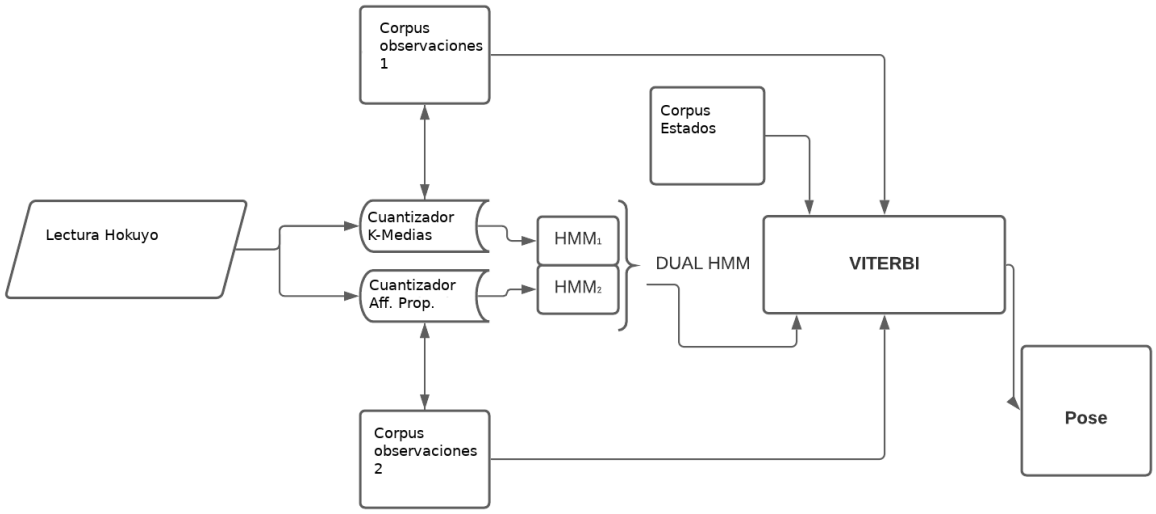


Figura 4.1: HMM-DUAL SLAM

En la figura 4.1 se observa en el primer bloque la entrada del sensor de tipo LIDAR Hokuyo sin embargo, durante la implementación se probó exitosamente el sistema usando información de tipo imágenes y nube de puntos, propios de los sensores **rgbd**, para el robot **HSR**. La información sin procesar del tipo 4.2 se cuantiza utilizando dos alfabetos, el primero utiliza el algoritmo de K-medias en su versión por lotes, y el segundo el algoritmo *Affinity Propagation*. A partir de la etapa de cuantización el flujo de información continúa de forma discreta, como una especie fusión de sensores para que mediante el algoritmo de Viterbi se elija, dentro del corpus de estados, la secuencia que mejor explique las lecturas de acuerdo con los modelos del bloque anterior.

4.2. Alfabetos de Observaciones ó Corpus de Observaciones

El sistema propuesto funciona con lecturas de un sensor Hokuyo, la lectura es un conjunto (más de 900) de mediciones de distancia distribuidas en un arco 270° frente al robot $\vec{O}_t = [R^t_0, R^t_1, R^t_2, ..R^t_{927}]$.

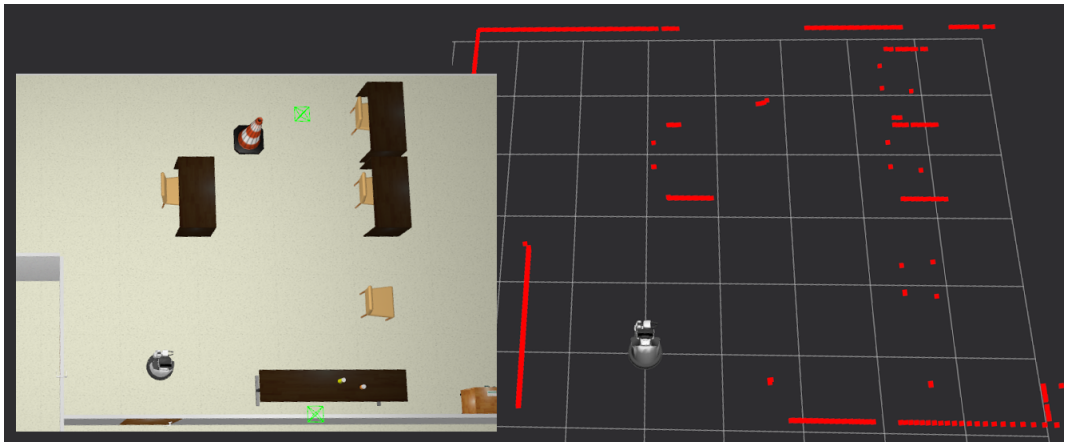


Figura 4.2: Una típica lectura del sensor Hokuyo utilizado por HSR.

El sistema propuesto es de naturaleza dual, por lo que la elección de los sensores y sus métodos de cuantización dependen en gran medida de las condiciones del ambiente, incluso la tarea a realizar [26]. La propuesta considera un par de cuantizadores, K-Medias (mini lotes) y affinity propagation (una adaptación de la técnica de reducción de dimensiones *Feature Propagation*) detallada en la sección 3.2.

De una forma alterna se reportan resultados de un vector de observaciones formado por puntos de interés o *landmarks* extraídos de una imagen **rgb** y algunas redes neuronales, concretamente un vector de mas de 2,000 características (*features*), formado por la última capa de una arquitectura **Resnet 50** [27].

Los resultados observados sugieren que los modelos se comportan de manera similar a el caso DUAL -con sensor Hokuyo-, sin embargo, el costo computacional se incrementa drásticamente al utilizar una red neuronal para extraer las características de la imagen. Lo que sugiere que la implementación es menos práctica, pero posible en caso de necesitar favorecer un **SLAM** visual. En [28] Dieter propone varias métricas de comparación para el éxito de las soluciones a **SLAM**. De forma que la elaboración de un mapa puede poner en perspectiva las decisiones necesarias para decidir que esquema de sensor o conjunto de sensores utilizar. Tal vez, las condiciones de luz, o de tráfico en el área, podrían imponer la elección del tipo de percepción a utilizar. Es importante considerar que cada entorno, incluso cada región dentro de el entorno, podría tener características que favorecerían a uno u otro enfoque y que una solución general justificaría la exploración de varios generadores de lecturas. Cabe destacar que la combinación de ambos, en un esquema de fusión de sensores, ha obtenido los

mejores resultados en general. Dado las tareas evaluadas, el robot en particular, la iluminación del ambiente incluso la cantidad de personas presentes, etc.

Por esta razón, se concluye simplemente que, dadas las aplicaciones exploradas en este trabajo, el costo computacional extra no hace una buena elección. Aún así el desempeño es correcto y por lo tanto puede encontrar una aplicación. Este razonamiento permite suponer lo mismo en caso de una implementación en 3d. Especial atención al trabajo de los compañeros del laboratorio en [29].

En el sistema de simulación propuesto (Gazebo) el sensor utilizado es un lidar Hokuyo. en la figura 4.3 se muestra una típica lectura del sensor Hokuyo, un vector de 720 lecturas distribuidas en un arco de 270 grados frente al robot. $\vec{O}_t = [R_0^t, R_1^t, R_2^t, ..., R_{720}^t]$

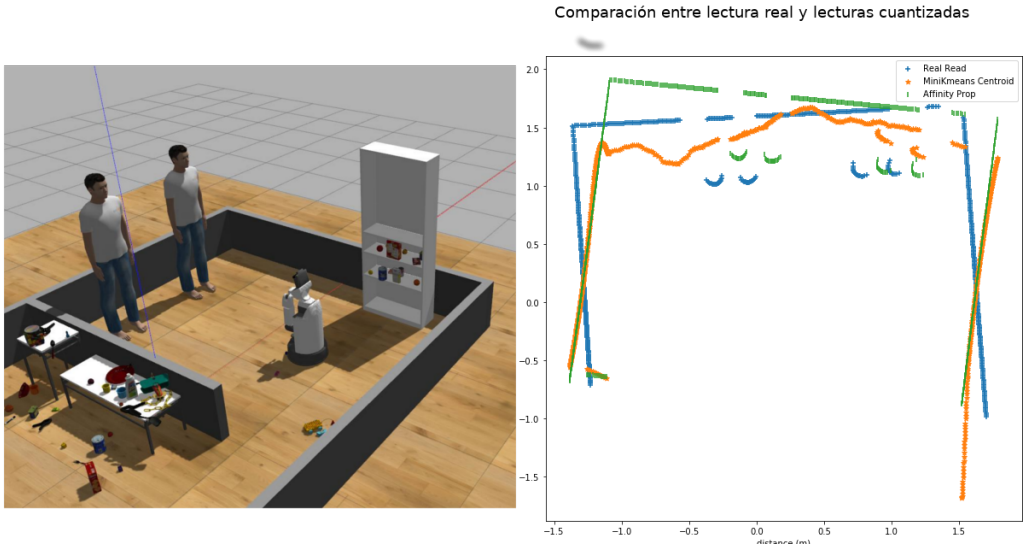


Figura 4.3: Una típica lectura del sensor Hokuyo utilizado por HSR (en simulación en azul, el ejemplar que la representa con Aff. Prop en verde y el centroide de K-Medias en naranja.

Ambos alfabetos son un tipo de cuantización vectorial, cada uno con un algoritmo de *clustering* propio. En el capítulo 3 se mencionaron brevemente cada uno de los métodos utilizados y se presentan a continuación algunos resultados y conclusiones.

4.2.1. K-Means

A medida que el conjunto de datos de entrenamiento crece se debe considerar que el algoritmo necesitará más tiempo para procesar la información y la calidad de la información se degradará. Esto se debe a la naturaleza del algoritmo, ya que los promedios no ponderados que generan los centroides, dan más importancia a los valores más repetidos. Lo que genera que se degrade la información de lecturas más interesantes porque se encuentran presentes en una menor proporción (debido a esto se conoce que el algoritmo tiene una naturaleza frecuentista).

Al analizar la lectura que se ve en la figura 4.4, podemos observar en naranja que el centroide (que es un promedio de todas las clases que agrupa y representa) no conserva muchas de las características *landmarks*.

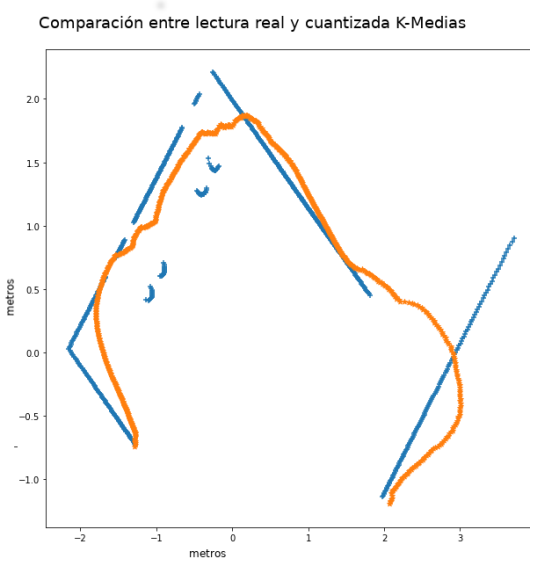


Figura 4.4: Centroide representando una lectura, con características disminuidas a raíz de la repetición.

A pesar de la obtención de resultados poco deseables, la agrupación de lecturas similares se realiza de forma correcta, como puede observarse en la figura 4.5 y gracias a la robustez inherente a su implementación, es dable asumir que es un método efectivo para resolver el problema de **SLAM** siempre y cuando sean considerados este tipo de inconvenientes.

Se puede aumentar el número de centroides (o símbolos del alfabeto) con el cual se representa el universo de lecturas, o se puede incluir un búfer de tamaño máximo de lecturas de entrenamiento. Cada una de las soluciones presenta ventajas y desventajas, finalmente representa un hiper parámetro que debe ser afinado en el sistema. En la sección de experimentos se propone un número de centroides tal que el valor de silueta (y por ende la degradación de la información) es prioritaria en comparación con el numero de dimensiones.

En la figura 4.6 se muestra tal ajuste en comparación con la figura 4.5

4.2.2. *Affinity Propagation*

Para mitigar la degradación de la información en los centroides, se propone contrapuntear con un alfabeto cuya naturaleza consiste en agrupar por similitud, es decir, sin crear un promedio. La propuesta consiste en elegir uno de los miembros del cúmulo, un "ejemplar representativo". El algoritmo es mucho más costoso computacionalmente, la implementación práctica se realizaría utilizando el mismo búfer mencionado en

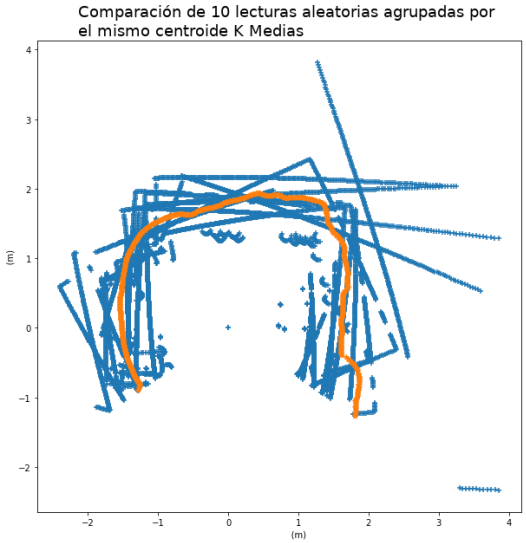


Figura 4.5: Centroide y 10 lecturas aleatorias representadas por éste.

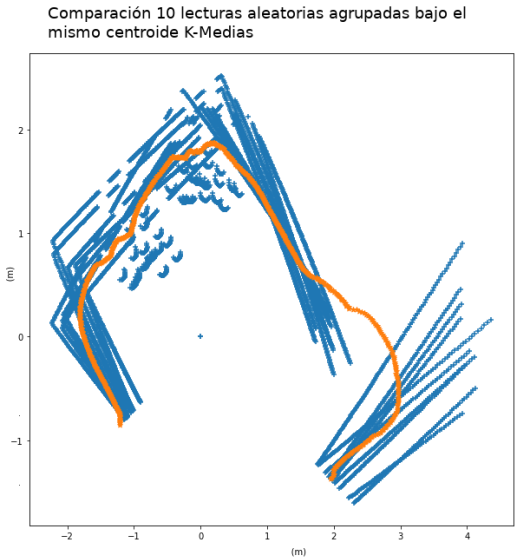


Figura 4.6: Centroide y 10 lecturas aleatorias representadas por éste, ahora con un mejor ajuste de número de símbolos y con un búfer finito de lecturas de entrenamiento.

el caso anterior y realizando un muestreo aleatorio limitarlo a un máximo de 15,000 lecturas del conjunto de entrenamiento.

En la figura 4.7 se observa la misma lectura que se uso en el ejemplo de la figura 4.4, pero ahora cuantizada con el algoritmo de aff. prop.

Los resultados obtenidos son congruentes con lo que se esperaba, es casi una contra parte del K-medias, conserva toda la información de las lecturas miembro del cúmulo

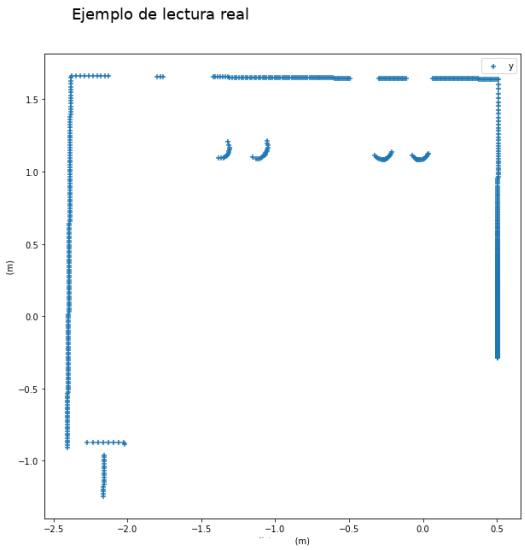


Figura 4.7: El ejemplar aff. prop. o símbolo que representa la lectura de la figura 4.4 con el alfabeto 2.

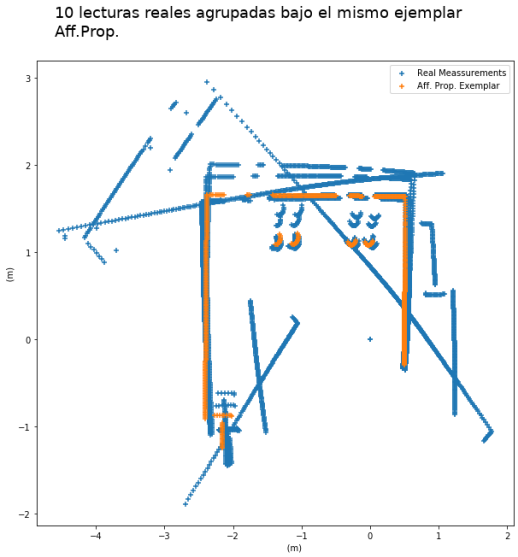


Figura 4.8: El ejemplar aff. prop. en naranja con 10 miembros aleatorios representados por ese símbolo en azul.

con el precio de mayor complejidad y más parámetros que ajustar. Al afinar los hiper parámetros del algoritmo podemos obtener cúmulos de lecturas más uniformes como se ilustra en la figura 4.9

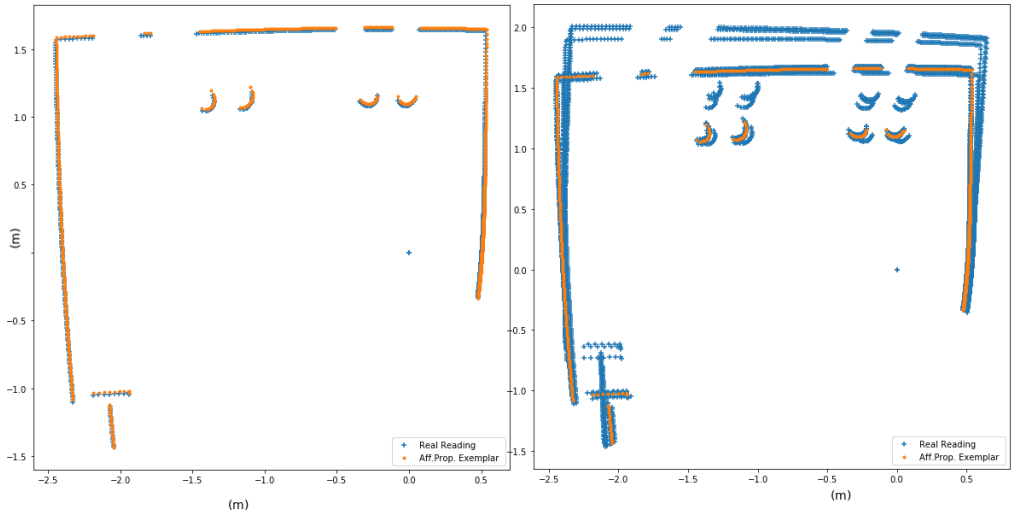


Figura 4.9: El ejemplar aff. prop. en naranja con 10 miembros aleatorios representados por ese símbolo en azul. Un mayor numero de ejemplares permite mejores cúmulos.

4.3. Estados y odometría.

El corpus de estados estará compuesto por las estimaciones de odometría encontradas en el conjunto de entrenamiento, existen diversas técnicas de agrupamiento,en los resultados que se muestran en la sección de experimentos se optó por agrupar los estados por su posición topológica y de acuerdo a un peso ponderado de la observación emitida. La obtención de valores de odometría es un factor relevante para la resolución del problema de **SLAM**, en el caso del robot **HSR** utilizado se obtuvo a partir de odometría de rueda y odometría láser. Es importante destacar que la operación se realiza sin mapa, es decir, no hay una estimación de una posición actual, simplemente se tiene una estimación de posición final, que se determina con base en la posición inicial y el conjunto de cambios entre cada paso de tiempo *timestep* como se muestra en la figura 4.11.

Odometría de Rueda

Se ha mencionado antes que es posible mantener un estimado de posición con la información de motores y de rotación de la rueda. Esta estimación aporta un factor de error que de incrementa en cada *timestep* hasta que deja de ser confiable (después de un período largo). En la figura 4.10 puede observarse la comparación de la odometría de rueda (en rojo) contra la posición real del robot (en azul), medida externamente. Es importante destacar que también debió realizarse una corrección dadas las condiciones iniciales, pues la odometría de rueda no utiliza ninguna referencia externa. El mapa por definición debe contener un origen del marco de referencia, para el caso de la estimación de odometría el origen siempre estará en las coordenadas cero (lectura

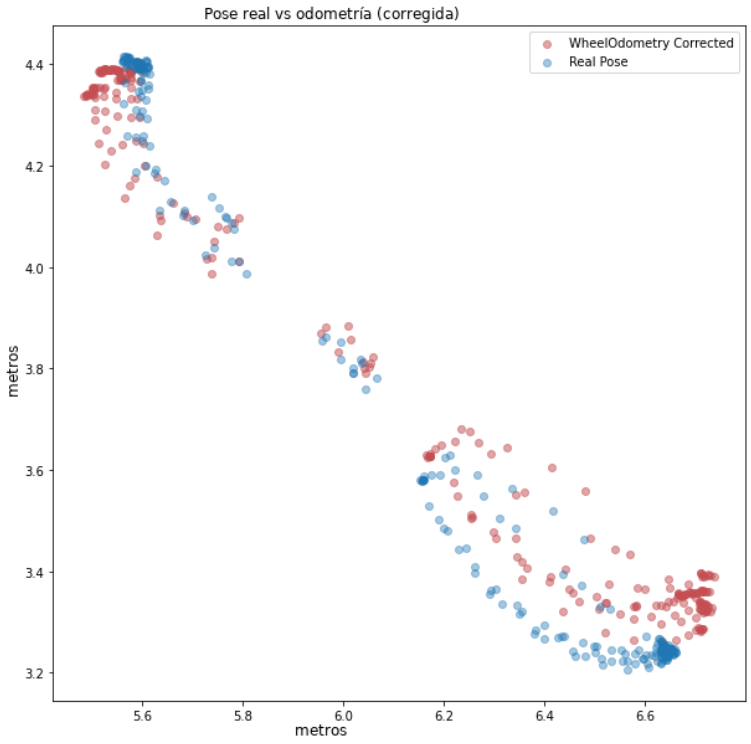


Figura 4.10: Una transformada inicial alinea las lecturas para permitir comparar el error, i.e. condiciones iniciales conocidas.

roja).

Odometría Láser

Existen aplicaciones donde la odometría de rueda no está disponible, p.ej. los drones, o interacciones piso-rueda cuyo error no satisface las necesidades de precisión de la aplicación. La mayoría de los algoritmos implementados en **ROS** utilizan alguna variante de la familia de algoritmos (*Iterative Closest Point*). Supóngase que el valor de referencia (lectura en rojo) en la figura 4.12, **ICP**, busca calcular la transformada rígida tal que minimice la distancia entre los puntos correspondientes, dicha transformada de traslación y rotación será una estimación de la delta de odometría, obtenida a partir del láser, por lo anterior se le denomina en la literatura como odometría láser.

Los algoritmos de transformada rígida tienen una complejidad computacional media, la premisa consiste en aumentar la eficiencia al minimizar la normal entre cada punto, no la distancia entre puntos. En la figura 4.14 se muestran las normales de la lectura del ejemplo de la figura 4.12 y el centroide relacionado al símbolo obtenido de la cuantización K-medias de esta lectura.

El tamaño del corpus de observaciones de K medias es menor a 512, por lo tanto el cálculo de las normales reduce a la mitad el algoritmo a utilizar y nos permite obtener odometrías corregidas dentro del espacio del estado. En la figura 4.15 se muestran los

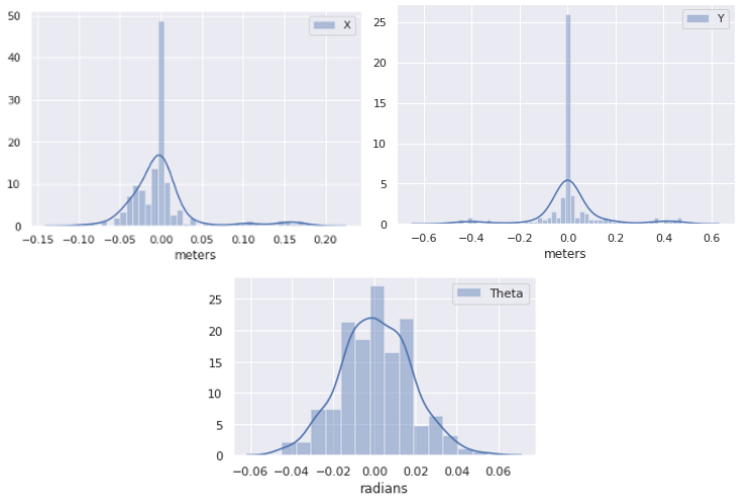


Figura 4.11: Caracterización de error en cada muestra, error absoluto en x, y y orientación theta. En todos los casos se observa una distribución gaussiana del error.

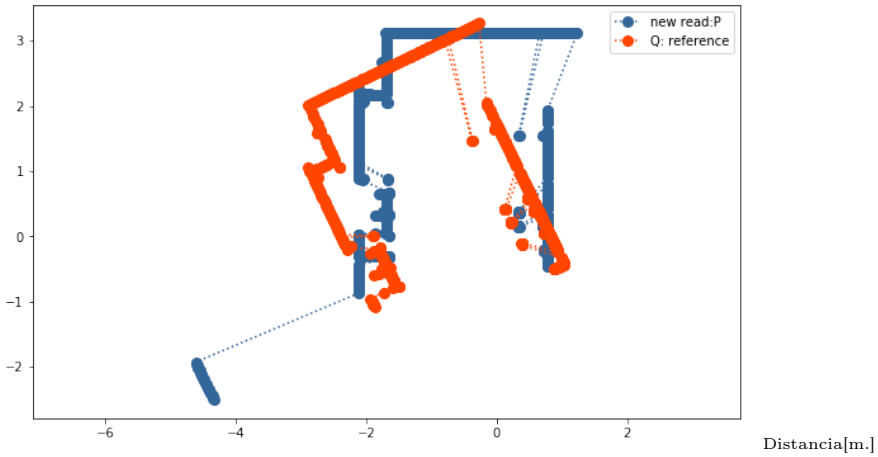


Figura 4.12: Comparación de dos lecturas en los tiempos t= n en rojo y t=n+1 en azul. Se busca encontrar la transformada rígida en 2 dimensiones que alinee mejor estas lecturas, un estimado de odometría láser.

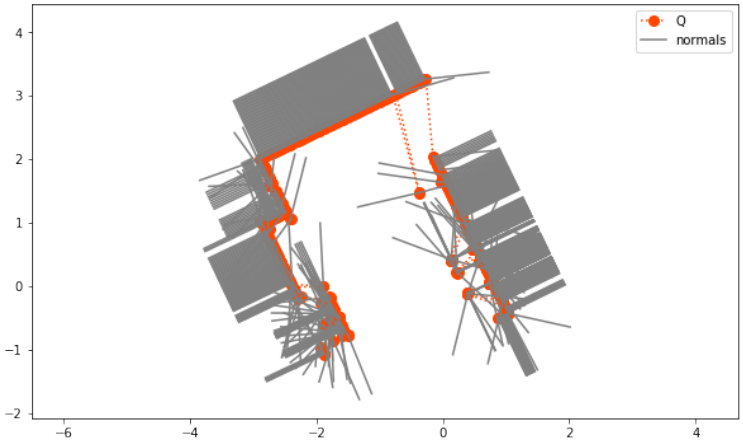


Figura 4.13: Normales de la lectura referencia.

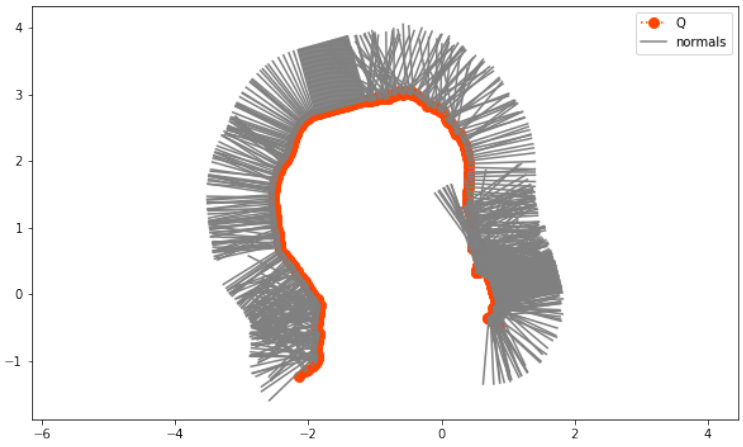


Figura 4.14: Normales del símbolo centroide relacionado a la lectura

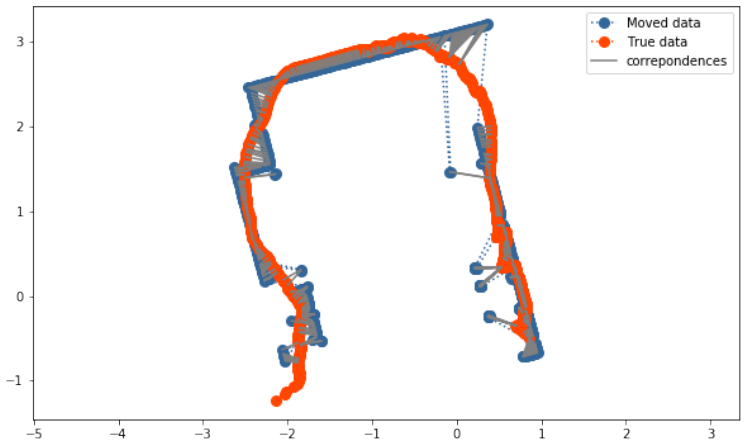


Figura 4.15: Lectura transformada con la traslación y rotación obtenidas.

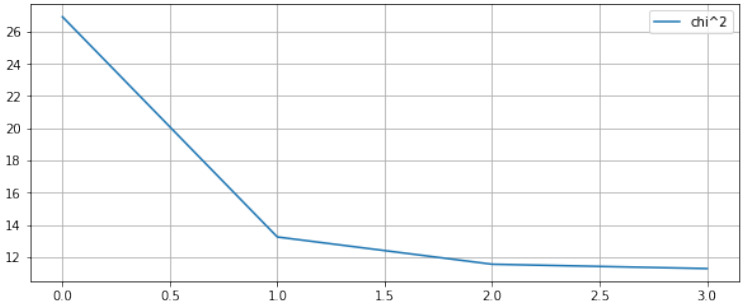


Figura 4.16: Minimización de las distancias permiten obtener transformada que alinea las lecturas de la figura 4.15

resultados de la minimización de las normales (azul) entre correspondencias obtenidas del símbolo (naranja). Los resultados numéricos de la minimización se reportan en 4.16. Esta nueva odometría láser permite corregir la odometría una vez que se estiman correctamente los estados del modelo.

Odometría probabilista

Al analizar los conjuntos de entrenamiento codificados por el sistema propuesto se observa que las transiciones entre estados ocurren, normalmente, en un punto bien definido. Normalmente en el sentido matemático de la palabra y con una distribución congruente con los resultados comentados en la figura 4.11.

Recordemos que se cuentan con ejemplos en los conjuntos de entrenamiento y que el cálculo estadístico de los mismos permite proponer una lista de la coordenada donde la transición tiene una mayor probabilidad de ocurrencia. Con base en dicha observación se sugiere una corrección en la odometría de rueda para restablecer las coordenadas a partir de dicha referencia. Gracias a la lista de coordenadas con mayor

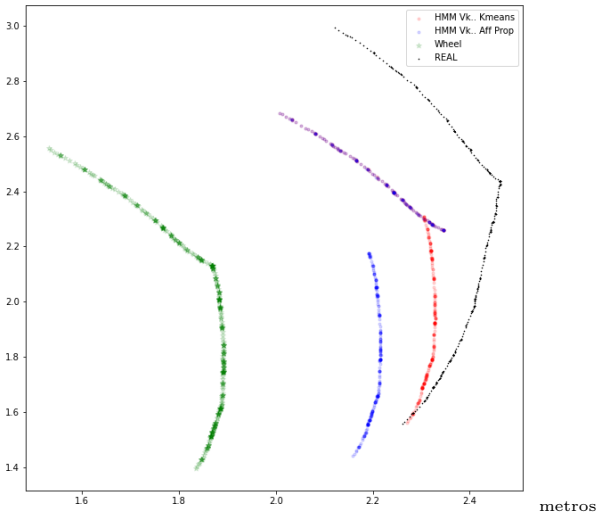


Figura 4.17: Comparación de odometrías

probabilidad de ocurrencia de una transición es posible mantener dos estimaciones independientes, y como se propone, una tercera que funciona mejor que cada una de ellas por separado, un **HMM DUAL**. En la figura 4.17 se muestra una serie de odometrías en el momento que el sistema propuesto detecta una transición entre estados, estimado de rueda (verde), la posición real (medida externamente en negro), una odometría mantenida con las observaciones cuantizadas con el alfabeto K-Medias (rojo) y la odometría mantenida con una cuantización utilizando (azul) *Aff. Prop.*

Considerando como parámetro prioritario la economía computacional se opta por el método de estimación de odometría sin embargo, en tareas relacionadas con el manipulador se sugiere utilizar el método láser porque los movimientos de base son necesarios para controlar algunos de los grados de libertad del efector y la alta precisión puede ser prioritaria en estos casos.

4.4. Entrenamiento Fuera de Línea

En los capítulos previos se han mencionado todas las partes necesarias para estimar un par de **HMM** mediante el algoritmo BAUM-WELCH, que en su versión *off-line* permite calcular los parámetros de los modelos con una elegante implementación de optimización por esperanza maximización **E-M** (por sus siglas en inglés, *Expecation Maximization*).

E-M es un proceso iterativo que permite estimar los parámetros de un modelo probabilístico con variables latentes u ocultas (como es el caso de los **HMM**) para maximizar su verosimilitud (*likelihood*), también llamado, similitud Máxima a Posteriori **MAP** (por sus siglas en inglés) propuesto por Dempster et. al. en [30]. En dicho artículo se modifica el algoritmo para ser utilizado por redes Bayesianas, una vez más, el caso de los **HMM**. A grandes rasgos cuenta con dos etapas, una de maximización y otra de calculo de verosimilitud (*likelihood*). Como se planteó en la ecuación 3.7 los

cálculos de verosimilitud *likelihood* $P(\vec{O} \mid \lambda)$ se pueden realizar de forma eficiente con programación dinámica.

El entrenamiento se realiza por episodios, es decir, en períodos temporarles que permitan confiar en la odometría de rueda (evitando el aporte del factor de error señalado previamente), generando así un conjunto de entrenamiento que formado por la tupla x, y, θ y su respectivo vector de lecturas.

El siguiente paso es conocido como etiquetado, presente muchos problemas relacionados con el aprendizaje automático *machine learning*. Consiste en la asignación de un estado u otro a cada tupla de posición y observación, un símbolo a cada vector de observaciones correspondiente de tal forma que el conjunto de datos sea transformado a uno compuesto por dos variables discretas, X el estado oculto, que representa a las posicones del robot y V un símbolo que representa la lectura del sensor o fusión de sensores (según sea el caso). Para la investigación reportada en esta tesis se exploraron varias estrategias de etiquetado sin embargo, una vez más, la naturaleza del problema de **SLAM** podría beneficiarse de algún otro método bajo ciertas condiciones del terreno y de información e incluso de acuerdo a la aplicación. Lo anterior corresponde a otro de los hiper-parámetros que afinar en el modelo general **HMM-DUAL** propuesto. En la figura 4.18 se observa una representación gráfica de todas las posiciones que componen el conjunto de entrenamiento. El contraste de la gráfica permite observar que las líneas más nítidas corresponden a las observaciones que ocurren con mayor frecuencia, aquellas que tienen que están relacionadas con referencias físicas utilizadas como puntos de partida en los episodios de entrenamiento. El conjunto de datos de entrenamiento establece una observación por cada uno de dichos puntos.

La figura 4.19 codifica con color el estado al que pertenece cada punto, y las flechas verdes representan el centroide que ha de representar los puntos de cada color.

4.5. Entrenamiento en línea (Optimización de Grafo)

Los modelos parametrizados pueden utilizarse para navegar, concretamente la matriz de transiciones A común a ambos **HMMs** del HMM-DUAL puede interpretarse de forma topográfica y con un grafo de conectividad basado en las distancias entre grafos. La planeación de ruta se implementa con el algoritmo de Dijkstra [31], el control de acciones de dicha navegación se implementa mediante campos potenciales [32], una especie de algoritmo de aprendizaje que sólo favorece la recompensa inmediata, de forma similar a un insecto atraído por la luz, incapaz de pensar a futuro (por ejemplo para rodear el objeto), existe el peligro de que la navegación permanezca en mínimos locales. Todos estos problemas ocurren en la implementación, pero quedan fuera del alcance de este trabajo.

Se realiza trabajo futuro con miras a una implementación de **RL** que permite mitigar estos problemas, pero para la experimentación se utilizó un paradigma de campos potenciales simple [32]. En términos de *reinforcement learning* es un algoritmo TD(0) el algoritmo con miras de recompensa a futuro TD(λ) sugerido como trabajo

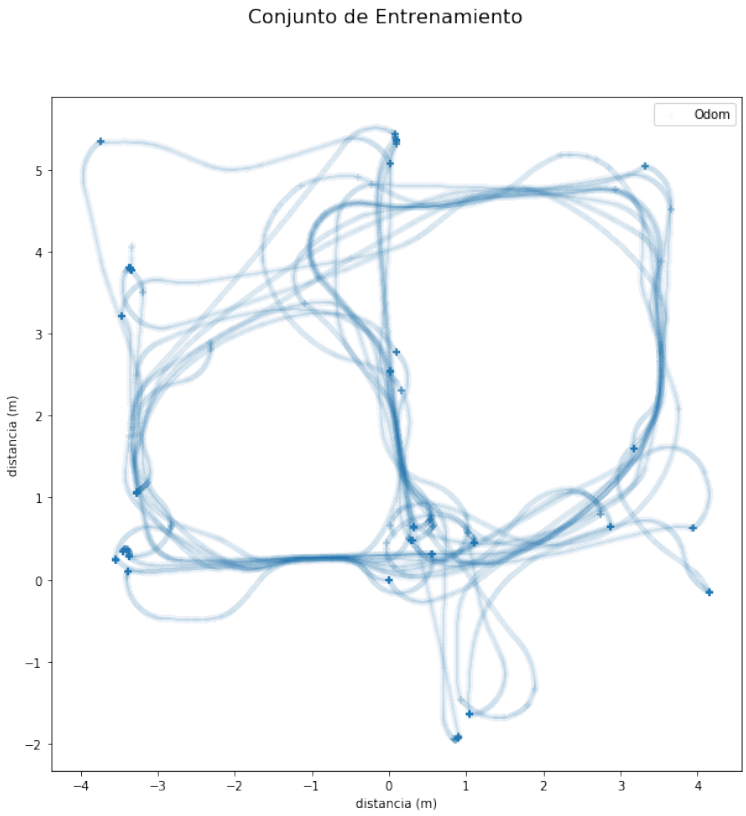


Figura 4.18: Conjunto de entrenamiento

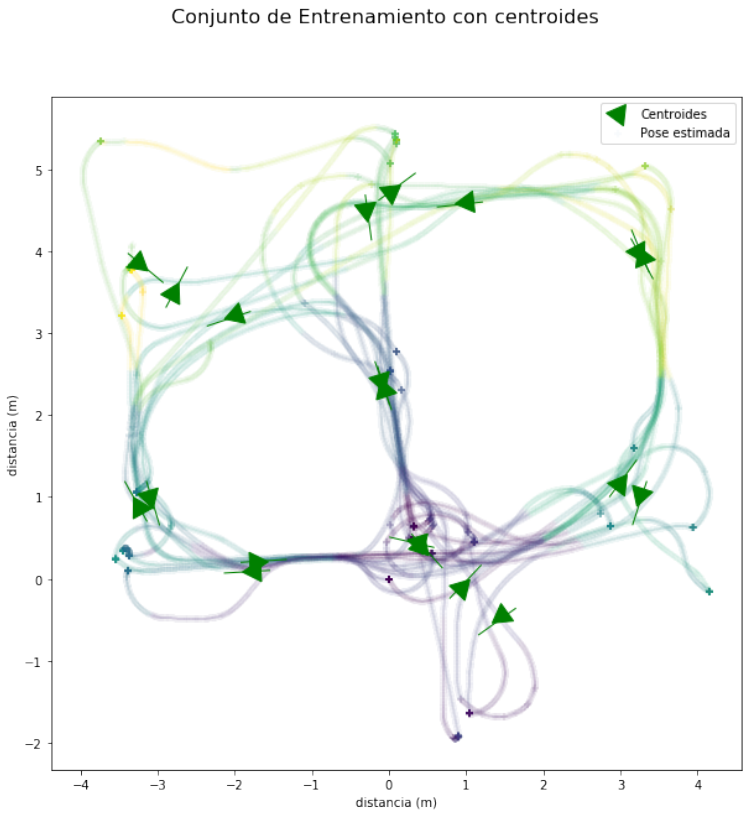


Figura 4.19: Cuantización de las poses que conforman el conjunto de entrenamiento 4.18, se observan en verde los centroides que representan a las lecturas (la pertenencia a uno u otro cúmulo se representa con el color).

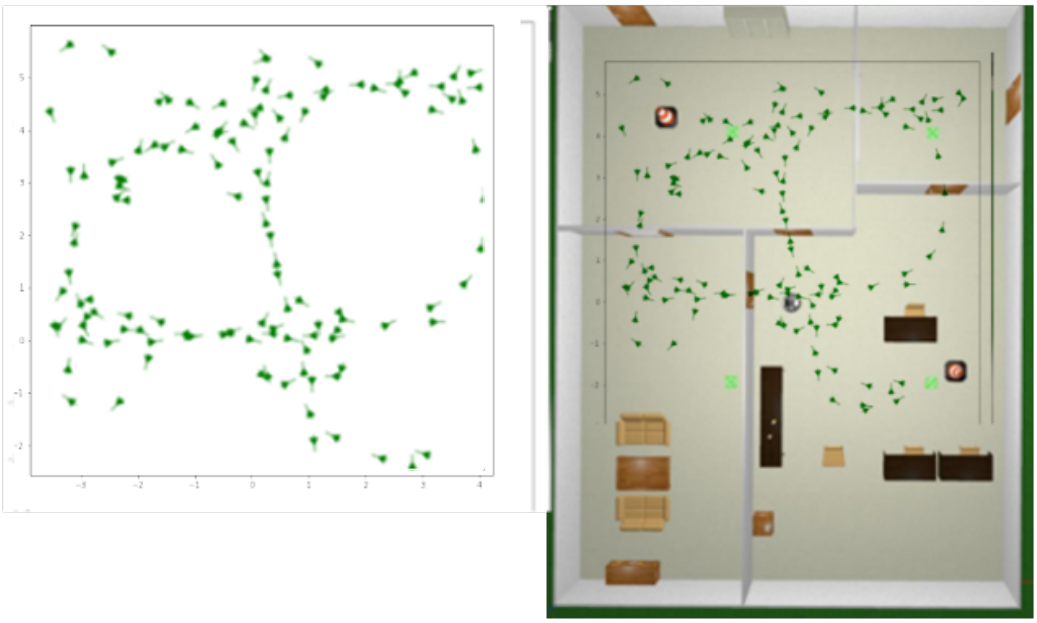


Figura 4.20: Representación topológica inicial del modelo obtenido fuera de línea

futuro muestra resultados muy interesantes.

Al cumplir con las condiciones descritas previamente el sistema será capaz de navegar y planear rutas a través de la representación topológica estimada fuera de línea; ahora el sistema tiene la capacidad de calificar cuando una navegación ha sido exitosa, es decir, llega a la meta. Los episodios de éxito se utilizan para implementar la mejoría en línea del modelo. Mediante el uso de las ecuaciones 3.1.2 y 3.1.2 se observa que la optimización del modelo depende solo del estado y la observación actuales, dichos parámetros sólo se modifican en dirección del gradiente y son ponderados por un valor η de taza de aprendizaje. Lo anterior significa que serán afectados en un valor diferencial el renglón y columna correspondientes a cada estado así como la observación, el ajuste a largo plazo del grafo favorecerá los parámetros que generaron corridas exitosas. En las figuras 4.20 y 4.21 se observa la optimización de la representación topográfica.

Es posible concluir que ambos métodos son equivalentes ya que el modelo converge en valores similares sin importar cuál de los dos se utilizó. Se obtendrían resultados similares entrenando al sistema mediante el uso del algoritmo de Baum-Welch que si se hubiese partido de un par de matrices con valores aleatorios y dimensiones que reflejaran el número de estados y observaciones que se desea estimar con el fin de calcular, episodio por episodio, los parámetros del modelo λ.

Finalmente, como se muestra en la figura figura 4.22, la navegación del mapa se realiza utilizando atractores virtuales en la secuencia de centroides que forman la ruta calculada por el algoritmo Dijkstra.

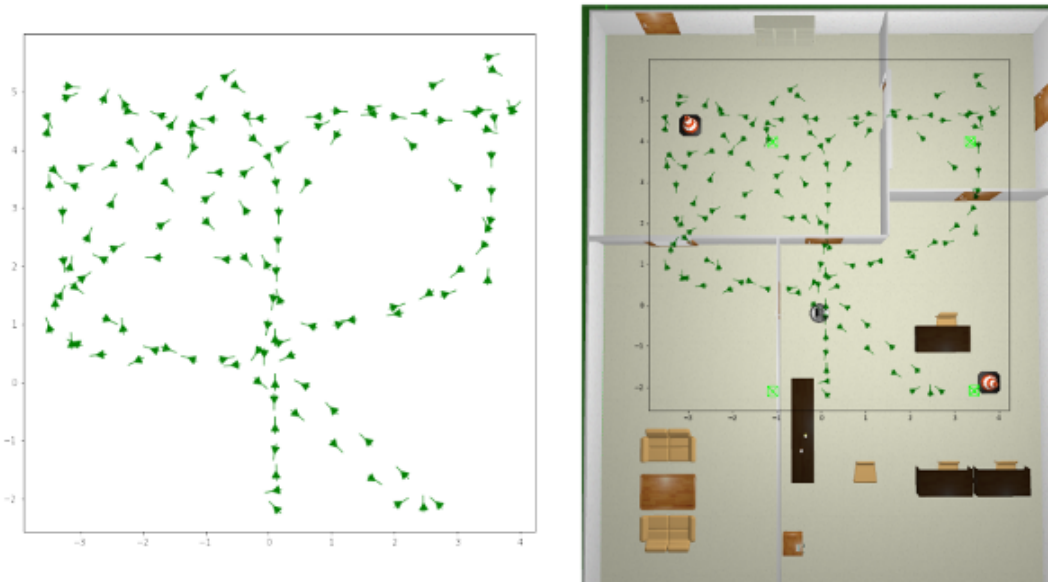


Figura 4.21: Una representación más estructurada después de varios entrenamientos en línea exitosos.

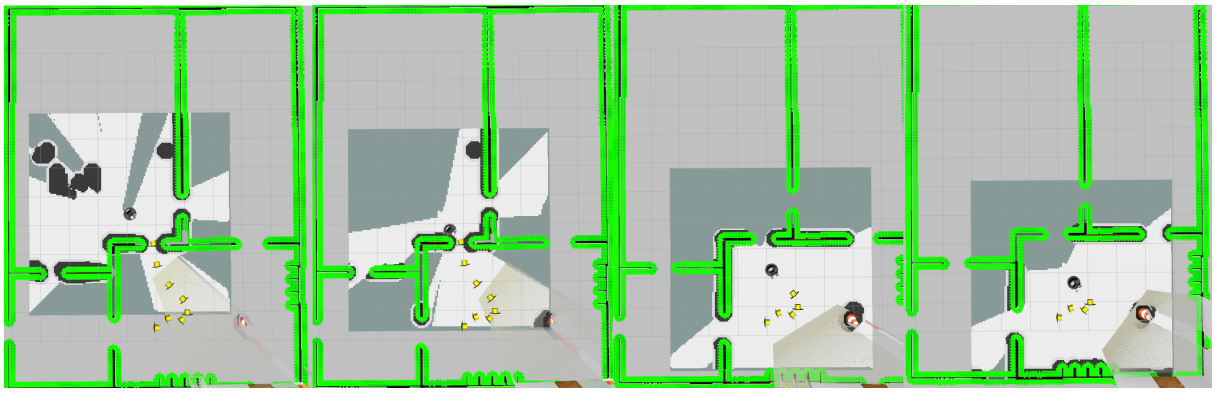


Figura 4.22: Varios momentos de la navegación de la ruta formada por atractores virtuales.

5 Experimentos

5.1. Planteamiento y evaluación

En este capítulo se documentan algunos de los experimentos utilizados para probar el sistema propuesto, además de algunos casos de interés para el trabajo futuro recomendado. Como se ha establecido a lo largo de este trabajo, la tarea de evaluar la solución de **SLAM** es algo subjetivo y con dependencia en factores como recorrido, ruta, aplicación, información previa, etc. Por lo que se buscó probar varias facetas relacionadas a **SLAM**.

Se propusieron experimentos con el paradigma de robot secuestrado [4], como se mencionó antes, la premisa radica en la omisión de condiciones iniciales y los estados son equiprobables, conforme el robot acumula observaciones en el búfer de Viterbi, los modelos producen estimaciones más confiables.

Los modelos se evalúan utilizando la misma metodología que se utiliza con los algoritmos de clasificación dentro del *machine learning*. Del conjunto de datos se determina un subconjunto, el de datos de entrenamiento que se utilizará para estimar los parámetros para los **HMM** del **HMM DUAL**. Posteriormente, el subconjunto restante se utilizará para comprobar que el modelo sea capaz de generalizar con ejemplos pertenecientes al conjunto de prueba o *test set*, de corridas nunca vistas durante el entrenamiento. La elección de los elementos del conjunto de datos es aleatorio, y en congruencia con la ley de los grandes números, no depende de los elementos de los subconjuntos, el modelo debe ser siempre aproximadamente el mismo.

Esa métrica de evaluación permite afinar los hiper parámetros del modelo y consiste simplemente en la estimación correcta de las secuencias de estados, variable oculta del modelo, sin considerar los métodos de estimación de odometría, descritos en 4.3. Se utiliza una ponderación que refleja el paradigma de robot secuestrado, de forma que se asigna un peso mucho mayor a la correcta estimación de los últimos 5 estados del búfer de Viterbi, mientras que se asume que los primeros estados son un sorteo equiprobable de todos los estados (Condiciones iniciales desconocidas). En la figura 5.1 se muestra una representación gráfica de una corrida del conjunto de prueba. Se asignó un color rosa al inicio de la corrida, cuando las estimaciones se realizan sin un buffer de Viterbi completo, y por ende se ponderan poco en comparación de las estimaciones finales representadas en amarillo. Los estados reales se representan en azul mientras que los estimados en rojo. En el fondo de la imagen se aprecia aún el conjunto de entrenamiento, así como todo el corpus de estados en verde.

Si bien este procedimiento permite hacerse una idea de que tan fielmente el modelo

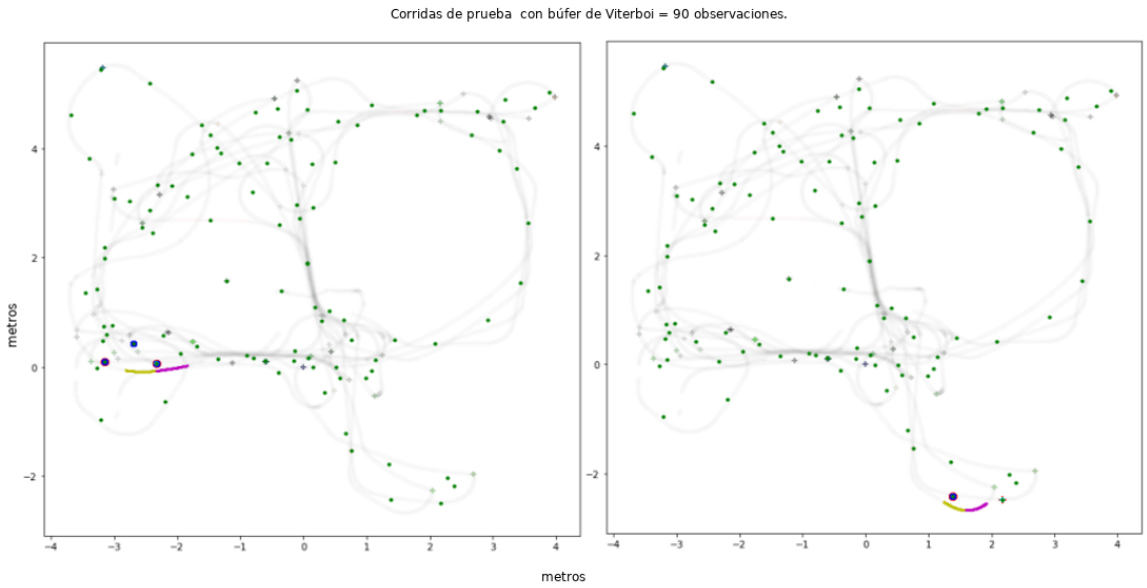


Figura 5.1: Dos corridas del conjunto de datos de prueba con sus respectivas estimaciones. Utilizando un búfer de Viterbi de 90 lecturas.

fue capaz de representar el conjunto de datos, no es una métrica útil en términos **SLAM**, pues se observa que el grado de éxito en la estimación depende más de la zona del mapa donde se toma la muestra que del modelo en sí. Por eso se propone una métrica de comparación de la pose estimada a partir del modelo completo con la pose real, y los experimentos ahora consisten en episodios supervisados de navegación y localización. Se busca evaluar no sólo la precisión de los modelos como en el caso anterior, sino su capacidad de resolver **SLAM**.

En casos donde se reporta una verdad absoluta, o pose real, las mediciones de la misma fueron obtenidas externamente, a veces con cinta métrica y referencias físicas, o utilizando mapas y algoritmos de localización, concretamente **AMCL**, sólo como una línea de base para poder medir el desempeño del sistema. La odometría de rueda es utilizada como otra medida de comparación, además de para implementar el método de corrección de odometría probabilista comentada en la sección 4.3.

5.2. Modelos

En esta sección se reportan algunos hallazgos interesantes al explorar varias arquitecturas de estimadores. A continuación se muestra un modelo formado por la fusión de la medición del Hokuyo y el vector de características extraído de una red neuronal convolucional **CNN** (por sus siglas en inglés, Convolutional Nueral Network) con arquitectura Resnet 50. Es importante resaltar que debido a la diferencia de escalas entre ambas lecturas se realizó un pequeño acondicionamiento de datos. Los resultados muestran un desempeño nominalmente superior al **HMM-Dual**, por lo tanto se

decidió no utilizar el método ya que se incrementa el costo computacional al utilizar una red neuronal, con un número de parámetros de alrededor 23 millones, contra los centenares o miles que típicamente bastan para la HMM-DUAL.

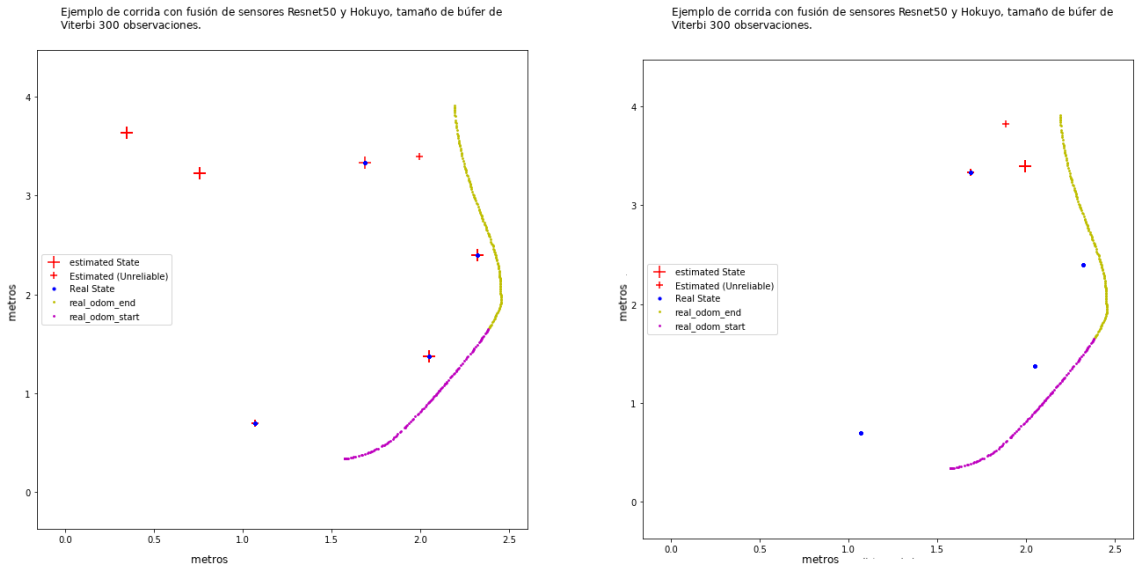
El modelo propuesto sólo requiere leer la imagen de cada *timestep* y propagarla hacia delante de los pesos obtenidos de los autores de la red, no es necesario hacer un *fine tunning*. Sin embargo, dada la dependencia de un sensor de imágenes a la luz ambiental, se considera un costo computacional extra innecesario, aún así es un caso interesante para ambientes ricos en información visual, con iluminación constante o simplemente en pasillos con poca información relevante para el Hokuyo. En la figura 5.2a se muestra una corrida típica con una estimación correcta realizada simplemente con el Modelo **HMM** del Hokuyo. Puede apreciarse que, si bien, la estimación es exitosa, el modelo arroja muchos falsos positivos, y juzga el recorrido como uno mucho más largo de lo que es en realidad.

En contra parte, en la figura 5.2b se muestra el mejor desempeño del modelo con fusión de sensores. Puede apreciarse que las estimaciones más confiables (cruz grande) es en realidad la más cercana al punto final de la corrida. Debe observarse también que los estados iniciales , cuando el búfer no se ha llenado, son poco confiables dado el paradigma robot secuestrado, el modelo Resnet discrimina adecuadamente los estados poco confiables de la misma corrida.

En rosa el inicio de la corrida, en amarillo las muestras más recientes, en azul los estados que se encuentran en el conjunto de prueba y en rojo los estimados por el modelo. El tamaño del marcador de estimación denota la confianza que se tiene en la misma, una ponderación que como se mencionó antes, da cuenta del hecho de que las condiciones iniciales desconocidas presuponen estados iniciales equiprobables antes de llenar el búfer de Viterbi.

Entrenamiento por Lotes

Para el desarrollo de la propuesta reportada en esta tesis se utilizó la ponderación típica en la literatura del *machine learning*, *training/ validation /test split* 80 10 10. Los modelos resultantes al realizar un entrenamiento por lotes y dividirlos en épocas se compararon contra el enfoque BAUM-WELCH tradicional, se observaron modelos prácticamente iguales y se concluyó la hipótesis de que el entrenamiento episódico es funcional. Lo anterior permite optimizar los grafos en línea al realizar la supervisión de los episodios, por lo tanto se pueden seleccionar las lecturas de episodios exitosos para corregir los parámetros con la misma ecuación de descenso por el gradiente hasta converger, ver ecuaciones 3.1.2,3.1.2 y3.15. Los modelos resultantes al realizar un entrenamiento por lotes y dividirlos en épocas se compararon con aquellos que se obtuvieron utilizando el enfoque BAUM-WELCH tradicional, el cual requiere utilizar el conjunto completo de entrenamiento. Esto se observa directamente, pues los parámetros de los modelos obtenidos con cada uno de los métodos son prácticamente iguales (se observa a simple vista). Esto permite concluir que es posible estimar un modelo de forma episódica, utilizando muchas corridas de corta duración en el mismo ambiente (lo cual es práctico), y éste será muy similar a utilizar una sola corrida de larga du-



(a) Corrida de prueba con estimación exitosa del modelo Hokuyo con un HMM simple sin fusión.

(b) Corrida de prueba con estimación exitosa del modelo Hokuyo con un HMM simple con fusión.

Figura 5.2

ración (lo cual no es práctico). Esto permite considerar relativamente confiables las estimaciones de odometría de rueda

5.3. Estimación de Posición

Con base en el enfoque detallado en la sección 4.3 se recorren varios ambientes virtuales de manera reactiva manteniendo al robot en continuo movimiento con una navegación por campos potenciales. Las corridas se realizan en episodios, donde el robot repetidamente parte del mismo punto inicial para realizar sus tareas. La propuesta es que el robot pueda ir reconociendo (aprendiendo) su entorno mientras trabaja en otras tareas. En las competencias se utiliza la supervisión humana para preparar tareas como manipulación. Las lecturas y estimado de posición de rueda se almacenan para formar el conjunto de datos, previamente se cuenta con los alfabetos de observaciones y el corpus de estados, por lo tanto únicamente se almacena par estado / observación, donde ambos se cuantifican vectorialmente. Mediante el análisis fuera de línea de un conjunto de entrenamiento, al utilizar el algoritmo de BAUM-WELCH se obtienen nos modelos λ_1 y λ_2 con los alfabetos de K-medias y *Aff.Prop.* (respectivamente). Ambos modelos comparten la matriz de transiciones A a partir de la cual se propone el grafo topográfico propio del *Graph-Slam*. Aunado a lo anterior, al analizar el conjunto de datos de observación se calcula estadísticamente la matriz de poses de transición, donde se observa el valor con mayor probabilidad de ocurrencia en odometría durante la transición de un estado al otro, dicho valor corresponde a la corrección de odometría de rueda, como se mostró previamente en 4.3.

Los "mundos" utilizados para realizar las pruebas son ambientes de competencias internacionales como el **WRS** [33] y la *Robocup* [34], otros proyectos desarrollados por el autor de esta tesis en el laboratorio de Bio-Robótica de la **UNAM** han participado en dichos eventos que serán descritos a continuación.

5.3.1. Apartamento, mundo WRS2018

Para la **WRS2018** se definió un "apartamento" como arena virtual. Las dimensiones de dicho ambiente virtual, permitieron probar la modularidad del sistema.Para la prueba se propuso un Modelo **HMM-Dual** para cada habitación.

La localización se realiza con el algoritmo de adelanto, que permite elegir, primero el modelo de mayor verosimilitud (*likelihood*).

Una vez que se cuenta con una hipótesis de la habitación en que se encuentra el agente, se selecciona el modelo de dicha habitación y se procede, como de costumbre, con el modelo de máxima verosimilitud y Viterbi. Las figuras 4.21 y 4.22 muestran este entorno con su respectiva representación topológica, así como la navegación del grafo. Por otra parte, la figura 5.1 muestra dos ejemplos de estimación correcta en la posición, al menos en el contexto de espacio estados, adicionalmente se debe comparar cualitativamente el error de estimación, que se realiza con el algoritmo **AMCL** que hace las veces de un *ground truth*. Se debe especificar que no se pretende superar el desempeño de dicho algoritmo (tiene mucho más trabajo y tiempo de desarrollo), el método propuesto implica un costo computacional considerablemente inferior, al realizar la comparación es dable concluir que tienen un desempeño similar.

En la figura 4.17 se muestran estimaciones de odometría para un tiempo específico. En negro el *ground truth* calculado (obtenido externamente con **AMCL**), en verde la estimación de odometría de rueda (con errores parametrizados según la información obtenida de la interacción robot-piso en el laboratorio de Biorobótica en el año 2022), en Azul el modelo λ_2 que utiliza el alfabeto *Aff.Prop.* y en rojo la odometría mantenida por λ_1 y su correspondiente alfabeto (**K-Medias**).

Para ambos casos, después de un número grande de muestras (alrededor de 2 o 3 veces el búfer de Viterbi), los modelos tienen un mejor desempeño que la odometría de rueda, lo que sugiere un funcionamiento correcto de los dos sistemas propuestos por separado.

5.3.2. Arenas Tidy Up Room, WRS2020 Robocup 2021

La figura 5.3 muestra la arena de competencias internaciones **WRS 2020** y **Robocup 2021** la cuál también fue utilizada en el Torneo Mexicano de Robótica en 2021 (**TMR 2021**). En la imagen se puede observar en superposición el grafo topológico estimado mediante un **HMM-DUAL**. Dicha arena está disponible en los laboratorios de Tamagawa, se pretende mostrar que el modelo calculado en simulación, es capaz de cruzar el umbral de realidad (*reality-gap*) y navegar la contra parte real de esta arena.

De manera análoga al caso anterior, primero se comprueba la efectividad de cada uno de los modelos a partir del estimado inicial realizado fuera de línea con el

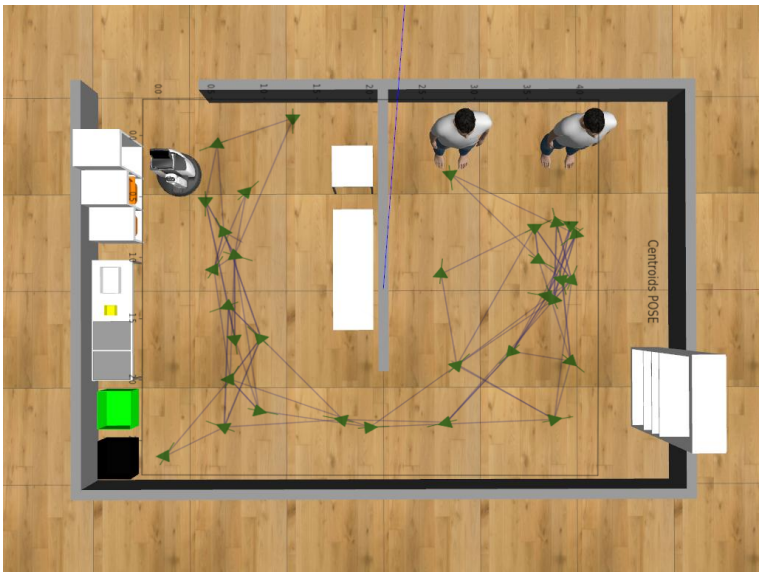


Figura 5.3: Arena utilizada en las competencias virtuales, con el grafo topológico asociado.

algoritmo Baum-Welch. Las figuras 5.5a y 5.5b, siguen las mismas convenciones de la figura 5.1 y muestran ejemplos de clasificaciones exitosas. La figura 5.4 muestra un ejemplo de una ruta navegación trazada por puntos virtuales de atracción que permiten recorrer la ruta trazada con campos potenciales para evadir obstáculos.

En el siguiente conjunto de figuras se presenta la comparación gráfica de cada estimador individual con el que se construye el **HMM-Dual** propuesto, en negro se observa el *ground truth* obtenido externamente por **AMCL**. La figura 5.6a muestra en verde la estimación de rueda, en rojo, naranja y azul las odometrías mantenidas por **HMM-DUAL**, **K medias** y **Aff. Prop.**, respectivamente. La odometría de rueda presentó mejor desempeño que los estimadores durante las primeras muestras de la corrida, incluso en los casos con información de estado inicial. La explicación para este resultado radica en que no hay una transición conocida del estado inicial (véase odometría probabilista 4.3). Al ocurrir la estimación del primer estado no se tiene un estado de procedencia y por ende la odometría se debe mantener a partir del centroide espacial. Lo anterior puede entenderse como la explicación del error de cuantización inicial. A medida que el agente sigue avanzando y recolectando lecturas los estimadores obtienen información suficiente para mantener la odometría a partir del valor de transición más probable con un error menor que el del centroide de cualquiera de los dos estados involucrados en la transición.

Al analizar esta corrida, pero en un período de tiempo mucho mayor, se hace más evidente la ventaja del sistema propuesto sobre la odometría de rueda. En las siguientes figuras únicamente se muestra el estimado mantenido con **HMM-Dual** en rojo contra la odometría de rueda en verde (con la finalidad de hacer más clara la comparación). Las figuras 5.7a muestran en verde las posiciones estimadas por la rueda, en rojo por el sistema propuesto, en negro las posiciones obtenidas externamente con

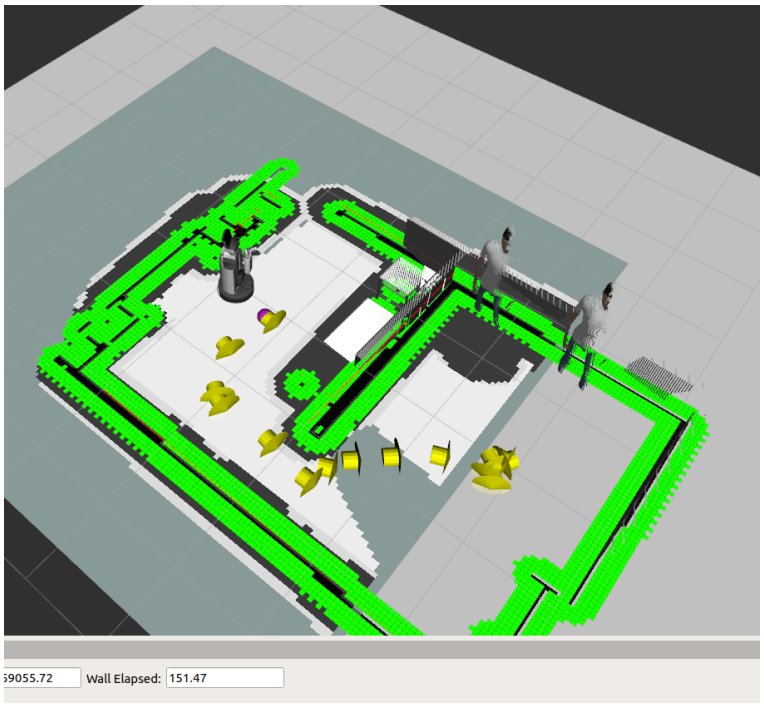
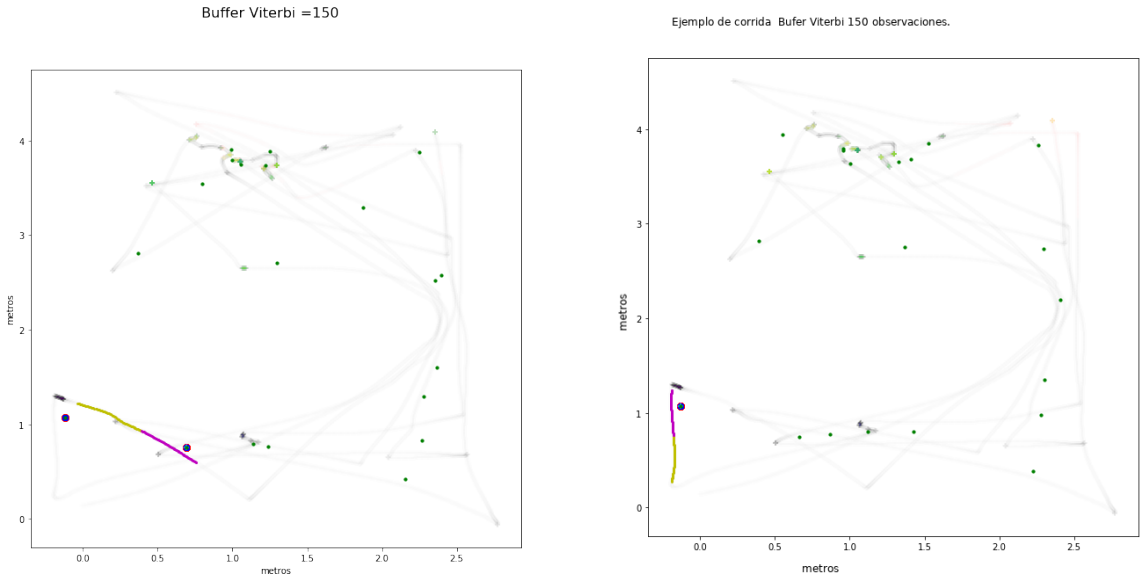


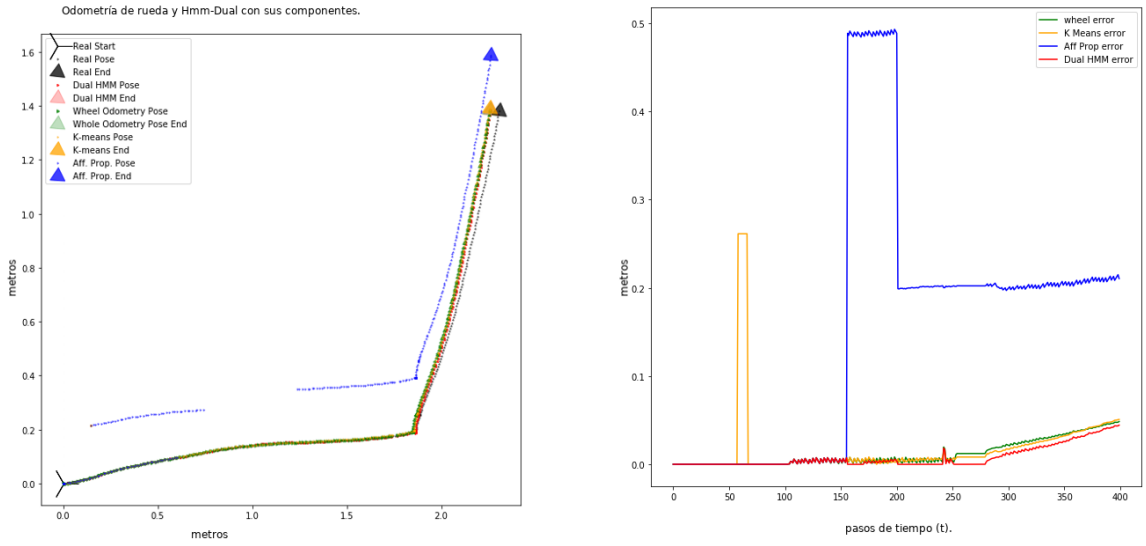
Figura 5.4: Ejemplo de ruta de navegación en el grafo topológico mostrado en la fig.5.3



(a) Ejemplo de corrida del conjunto de prueba con clasificación exitosa.

(b) Ejemplo de corrida del conjunto de prueba con clasificación exitosa.

Figura 5.5: Ejemplos de corrida del conjunto de prueba con clasificación exitosa.



(a) Poses

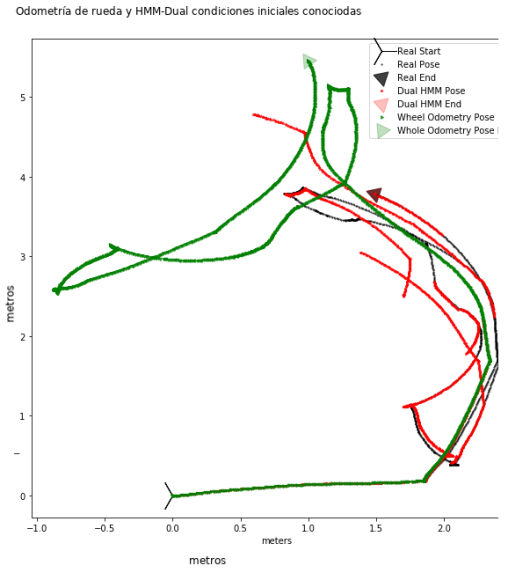
(b) Error absoluto de fig. 5.6a

Figura 5.6: Rueda vs. HMM-Dual vs. Real (condiciones iniciales conocidas).

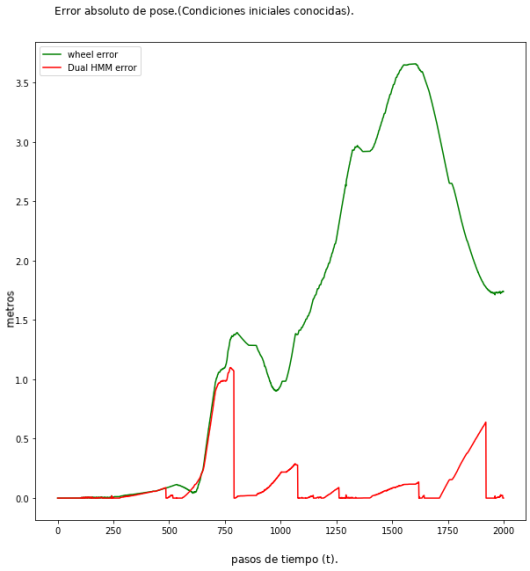
AMCL (*ground truth*); el error absoluto de estas dos odometrías comparadas contra el *ground truth* puede apreciarse en la figura 5.7b.

A continuación se muestra un ejemplo similar pero considerando el paradigma de robot secuestrado, es decir, sin condiciones iniciales. El error inicial de la odometría de rueda es un número aleatorio irreproducible, el resultado contrasta con **HMM-DUAL**, igualmente se muestran cada uno de sus componentes por separado en la figura 5.8a; la figura 5.8b muestra las magnitudes del error.

Las pruebas de la competencia presentaron un nivel de dificultad mayor ya que adicionalmente se debía evadir obstáculos menos evidentes, es decir, difíciles de detectar que pudiesen obstruir la rueda del robot, generando así una falla catastrófica. El experimento siguiente muestra una corrida que parte de condiciones conocidas, pero que al encontrarse con uno de dichos obstáculos se atora en la rueda del robot generando una falla catastrófica en la estimación de la odometría de rueda. El siguiente conjunto de figuras muestra el inicio de una corrida donde ocurre una falla catastrófica, el sistema es capaz de seguir funcionando y mantiene la prueba corriendo. Primero la fig.5.9a muestra el inicio de la corrida, antes de el evento, con su respectiva gráfica de error en la fig.5.9b. A continuación las figs.5.10a y 5.10b muestran el resultado durante evento con su respectivo error. Finalmente en las figs. 5.11b y 5.11a puede observarse el error y las odometrías después del evento catastrófico de odometría. En la figura 5.9b se observa que el error de cuantización antes de las 200 muestras es mayor que el de rueda (despreciable) al contar con condiciones iniciales. La figura 5.10a muestra el momento en que la odometría de rueda se pierde, arrastrando la estimación de **HMM-DUAL** hasta que detecta una nueva transición con éxito, corrigiendo la odometría con la odometría probabilística propuesta, como se planteó en

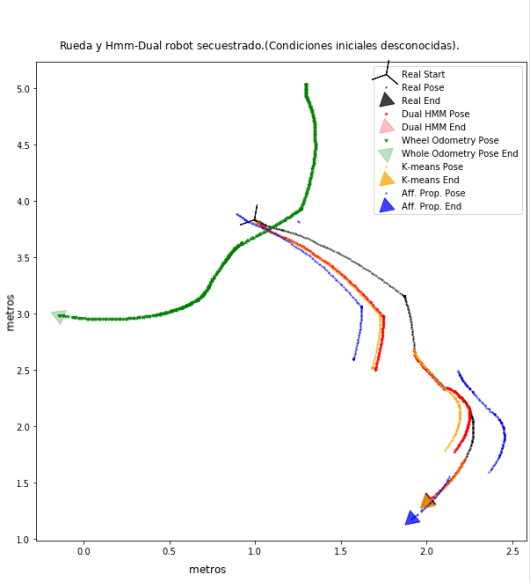


(a) poses

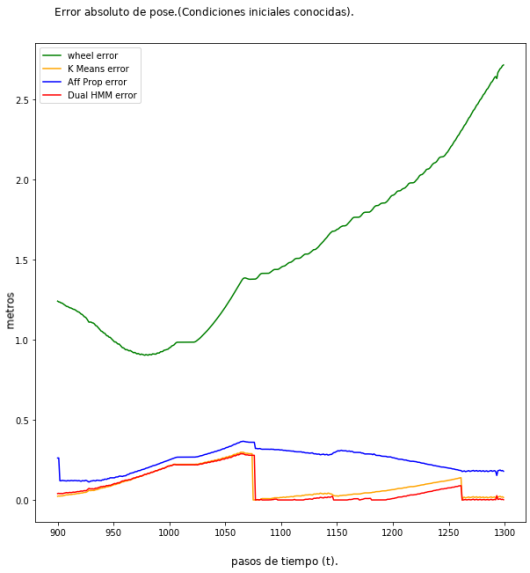


(b) Error absoluto de fig. 5.7a

Figura 5.7: La misma corrida de prueba de la figura5.6a pero en un intervalo de tiempo mucho mayor.

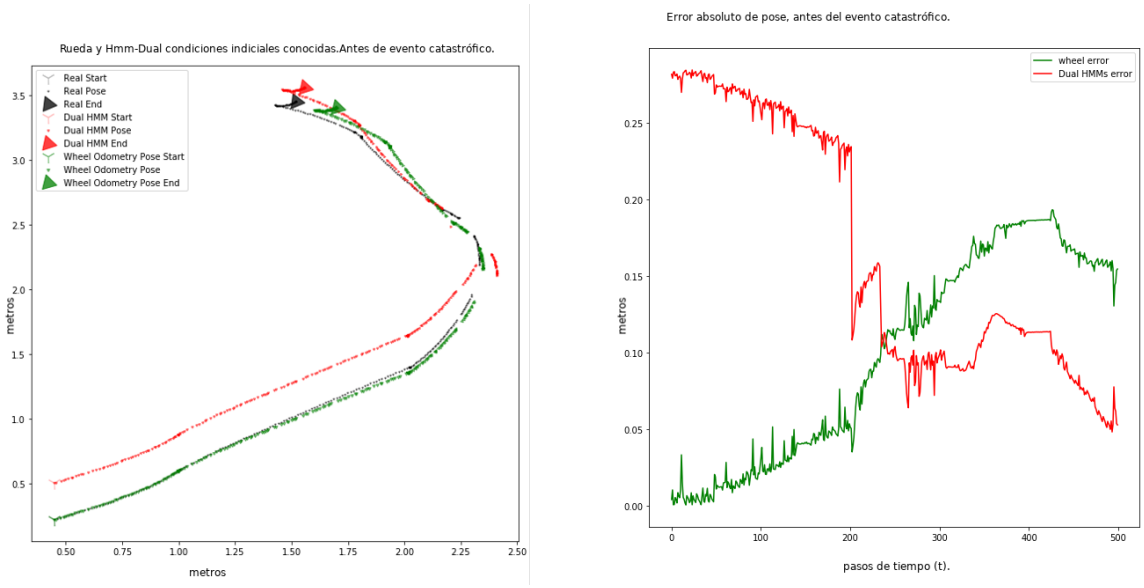


(a) Posiciones



(b) Error absoluto de fig. 5.8a.

Figura 5.8: Robot secuestrado,la odometría de rueda no cuenta con condiciones iniciales

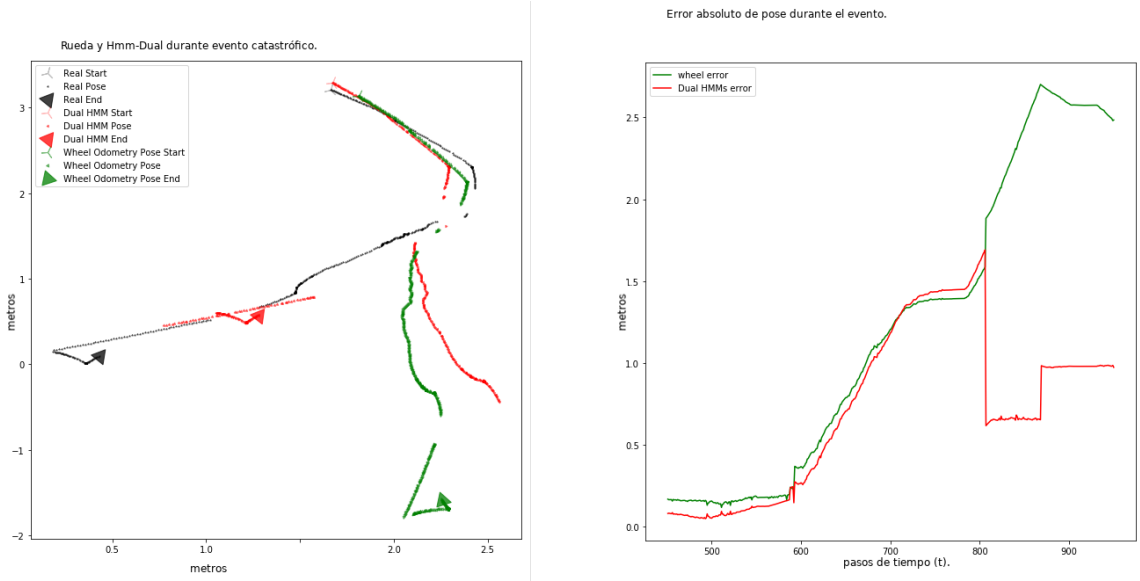


(a) Posiciones

(b) Error absoluto de la fig. 5.9a,

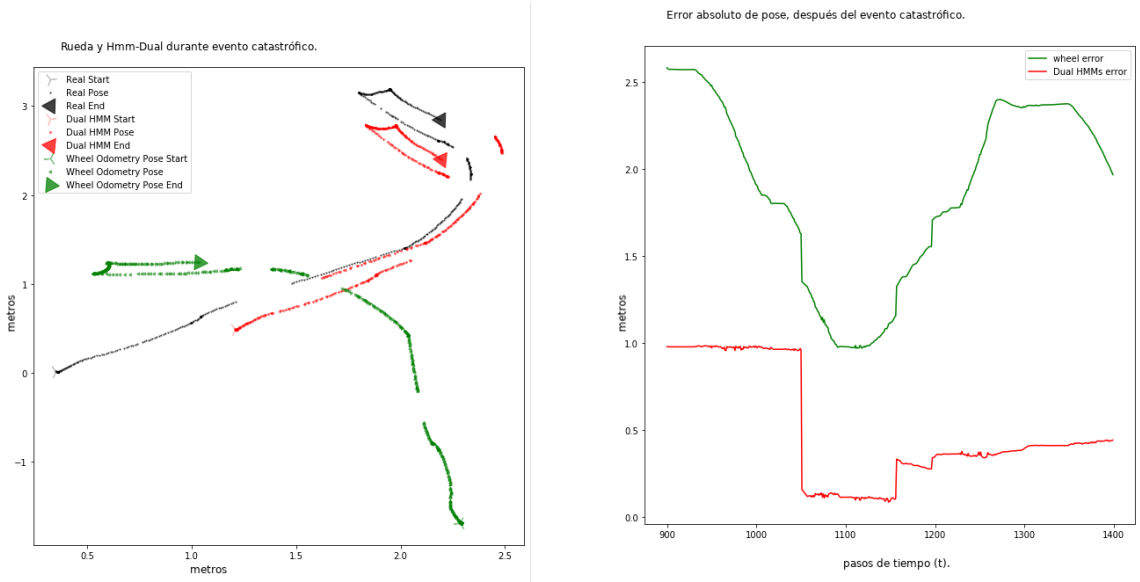
Figura 5.9: Antes del evento

4.3. Posteriormente se observa alrededor de la muestra 600 que el error se incrementa dramáticamente con el evento, como se aprecia en la figura 5.10b. Finalmente cerca de la muestra 800 ocurre una transición exitosa que permite reducir la incertidumbre del **HMM-DUAL**. Después de sucedido el evento catastrófico la odometría de rueda no tiene forma de relocalizarse, como se observa en la figura 5.11b, el error de rueda diverge. En contraste con lo anterior, el estimador **HMM-DUAL** mantiene la capacidad funcional sin perturbaciones como se muestra en la figura 5.11a). Al final de la corrida el **HMM-DUAL** tiene errores de magnitud aproximadamente iguales o menores a un radio del robot **HSR**.



(a) (b) Error absoluto de la figura 5.10a.

Figura 5.10: Durante el evento.

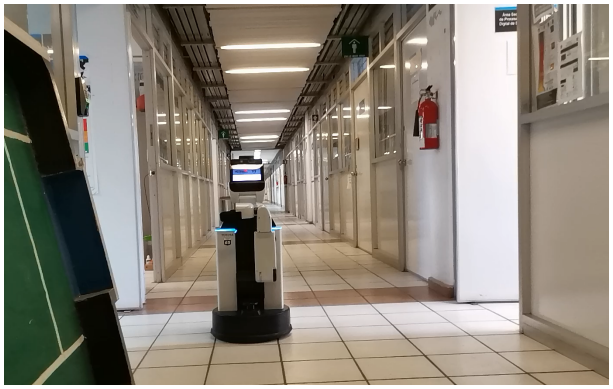


(a) poses (b) Error absoluto de la figura 5.11a.

Figura 5.11: Después del evento.

5.3.3. Laboratorio Bio-Robótica

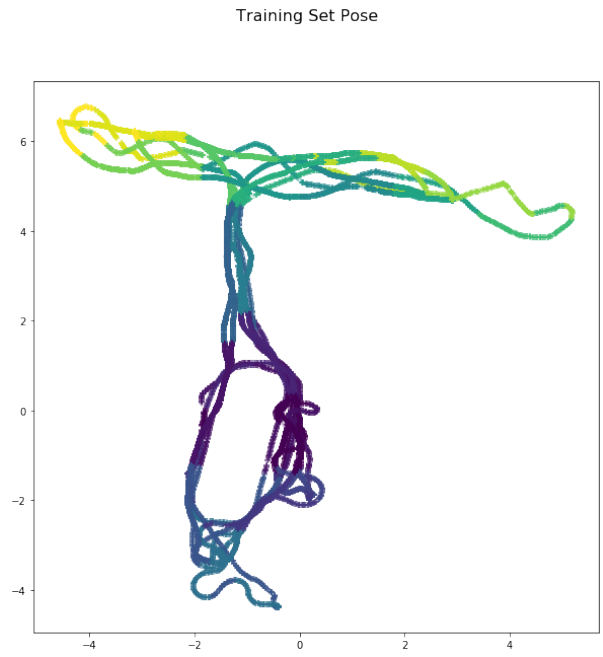
En esta sección se muestra un experimento similar a los anteriores pero realizado con el robot real en nuestro laboratorio. En las figuras 5.12a y 5.12b se muestran dos puntos del recorrido y al **HSR** recorriéndolo.



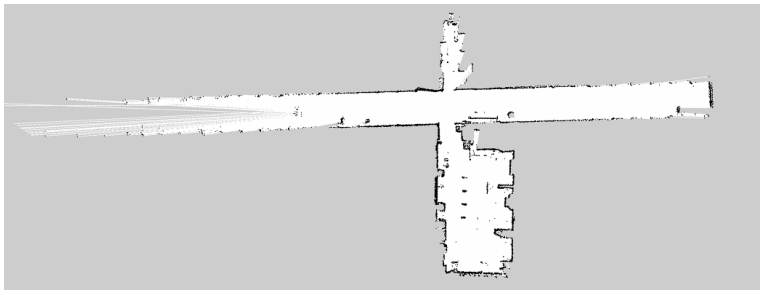
(a) **HSR** en el pasillo fuera del laboratorio de Bio-Robótica.

(b) **HSR** ingresando al laboratorio de Biorobótica.

Figura 5.12



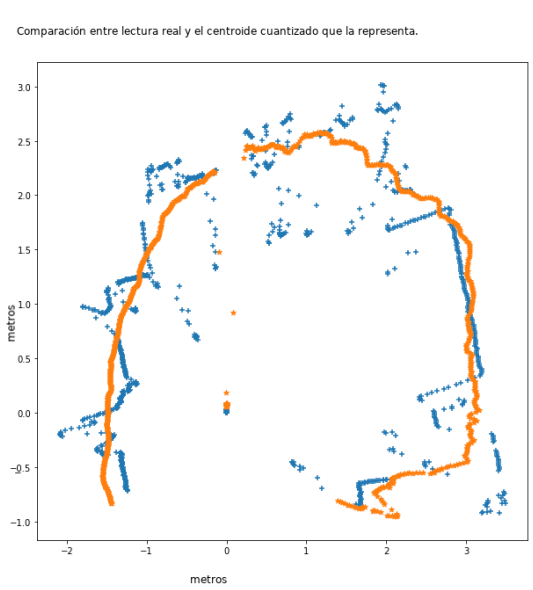
(a) Conjunto de datos de entrenamiento robot**HSR** en el laboratorio de Bio-Robótica de la UNAM



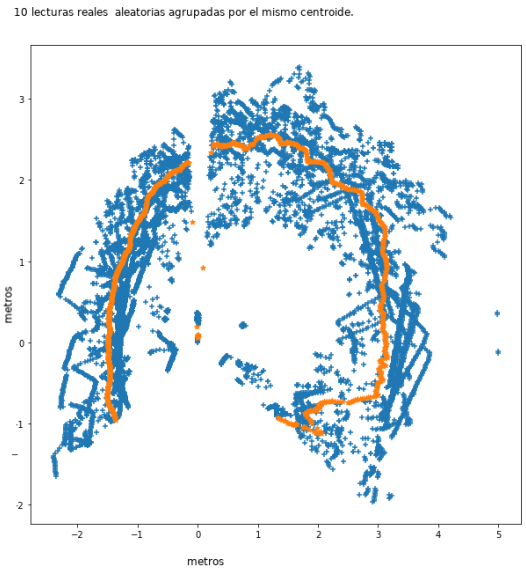
(b) Mapa de ocupación de referencia, no es usado por el método propuesto.

Figura 5.13

Las figuras que se muestran a continuación representan el desarrollo completo del experimento. Figura 5.13a el conjunto de datos de entrenamiento que se obtuvo con corridas episódicas, por lo tanto la odometría de rueda es confiable. Figura 5.13b el mapa que se obtuvo mediante *grid-mapping* clásico que se utiliza sólo como parte del los requerimientos de **AMCL** para el *ground truth* y para referencia visual del lector, sin embargo, el método propuesto no utiliza el mapa de ninguna forma, incluso el **ICP** sugerido en la sección 4.3 se realiza sobre el centroide o ejemplar de los símbolos.



(a) Lectura y su centroide



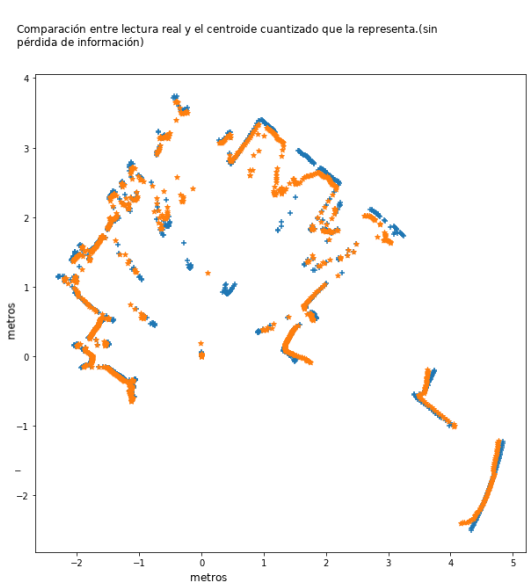
(b) Diez lecturas agrupadas bajo el mismo centroide.

Figura 5.14: Ejemplo de una lectura (en azul) codificada con un centroide que ha sufrido degradado de información.

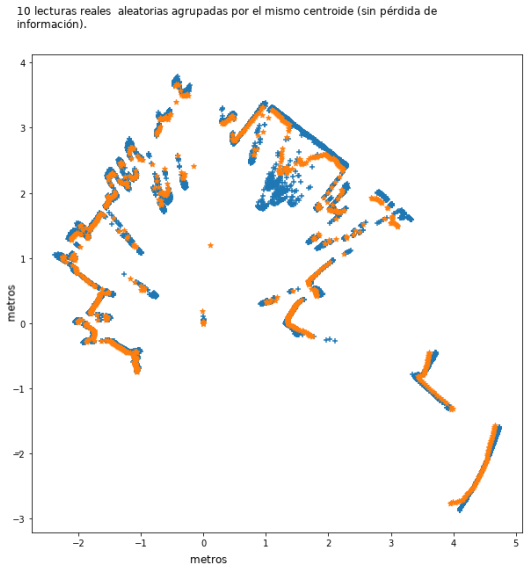
Las imágenes que se muestran a continuación son equivalentes a aquellas que se presentaron en la sección 4.2, particularmente el conjunto representado por las figuras 4.3 a la 4.9.

Primero, en la figura 5.14a se muestra un caso donde la información contenida en el centroide de **K-Medias** se ha degradado, por lo tanto el desempeño del algoritmo de agrupamiento no es muy bueno. Lo anterior puede observarse en la figura 5.14b. Las imágenes sugieren que el hiper-parámetro de número de centroides para **K-medias** debe cambiar (incrementar el número de centroides no necesariamente implica mejor desempeño).

En las figuras 5.15a y 5.15b se aprecia el caso contrario. En (a) se muestra un planteamiento similar al de la figura 5.14a, que corresponde al caso en el que se ha elegido un numero suficiente de centroides para la codificación; en (b) puede observarse en naranja el centroide y en azul 10 muestras de lecturas elegidas al azar pertenecientes al conjunto de entrenamiento, que son representadas por el mismo símbolo centroide. Al comparar con la figura 5.14b es evidente que tiene un mejor desempeño al agrupar lecturas. En la figura 5.16a se muestra la misma lectura del ejemplo mostrado en la figura 5.15a, pero representada en naranja por un ejemplar de *Aff. Prop.*, así como 20 muestras de lecturas elegidas al azar agrupadas para mostrar el desempeño superior al cuantizar correctamente. Finalmente en la figura 5.16b se observa el caso del pasillo en donde a pesar de tener un buen ejemplar podría ser necesaria una fusión de sensores con visión como se detalló previamente en la sección 5.2.

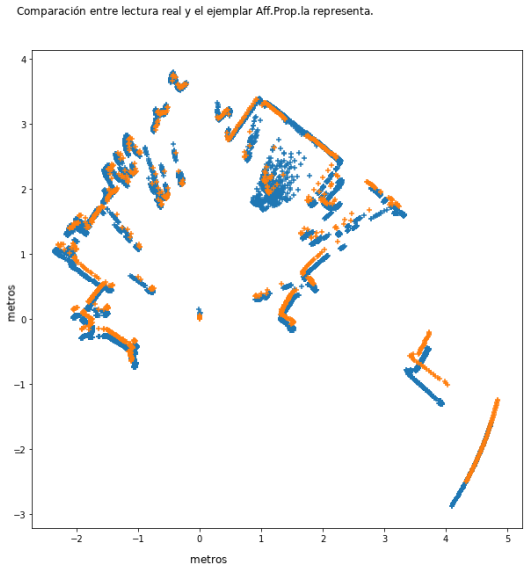


(a) Lectura y su centroide.

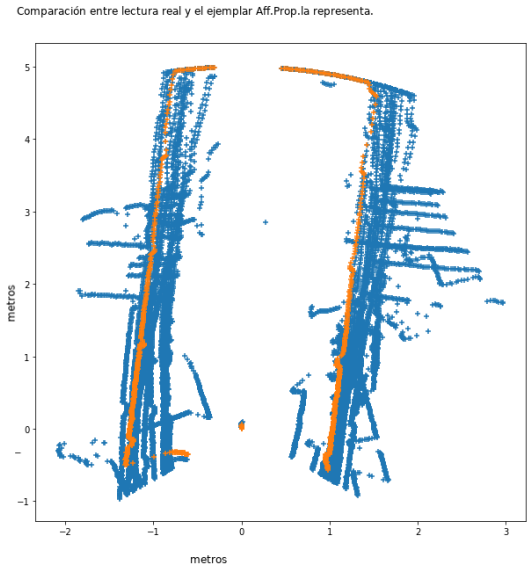


(b) Diez lecturas agrupadas bajo el mismo centroide.

Figura 5.15: Cuantización con K-Medias robot real.



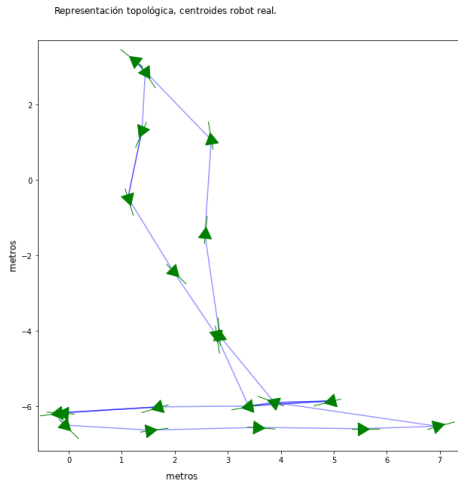
(a) Diez lecturas agrupadas bajo el mismo ejemplar.



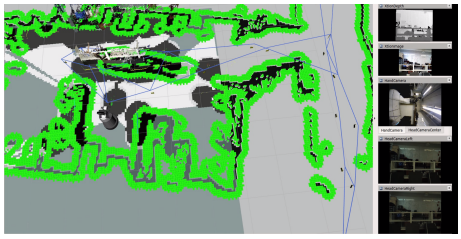
(b) Diez lecturas agrupadas bajo el mismo ejemplar.

Figura 5.16: Cuantización con *Affinity Propagation* robot real.

En la figura 5.17a se muestra el mapa topológico obtenido a partir de la matriz de transición del modelo **HMM-DUAL**. El robot es capaz de navegar este grafo y de ubicarse dentro del mismo con el algoritmo de Vitebri.



(a) Representación Topográfica de los modelos estimados con las mediciones de robot real en el laboratorio.

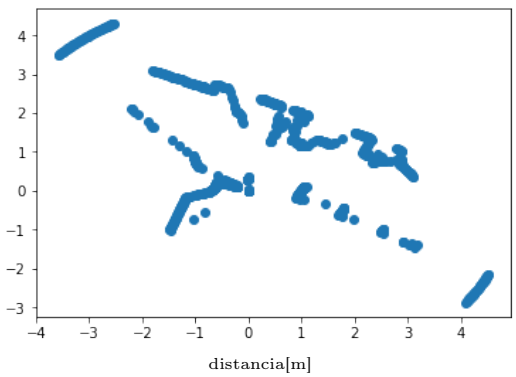


(b) Representación Topográfica de la figura 5.17a obtenida de las mediciones de robot real superpuesta sobre le mapa utilizado para obtener el *ground truth*

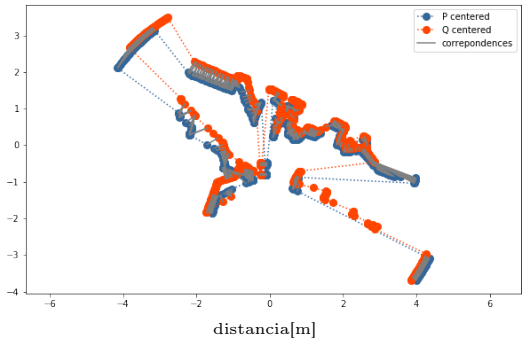
Figura 5.17

En las figura 5.1 de la sección 5 se mostró una representación gráfica para probar las estimaciones realizadas por el **HMM Dual** utilizando el conjunto de datos de prueba, con un subconjunto previamente seleccionado del conjunto de entrenamiento. Las figuras 5.14a a 5.16b representan la contra parte del modelo **HMM Dual** estimado y probado pero por el robot real en el laboratorio, fue necesario realizar ajustes mínimos, el tamaño de búfer y una tasa de muestreo ligeramente mayor. El color rosa al inicio de la corrida representa las estimaciones realizadas sin un buffer de Viterbi completo (poco confiables), en azul los estados clasificados exitosamente, en rojo las estimaciones menos confiables, en verde los puntos del corpus de estados. Se debe resaltar que las dimensiones del espacio explorado (poco estructurado) de aproximadamente 7x8 m ($56m^2$) corresponden al área promedio de un departamento en la Ciudad de México.

A continuación se muestra un ejemplo de estimación mediante odometría láser gracias a **ICP** considerando la lectura mostrada en la figura 5.18b se observa a la

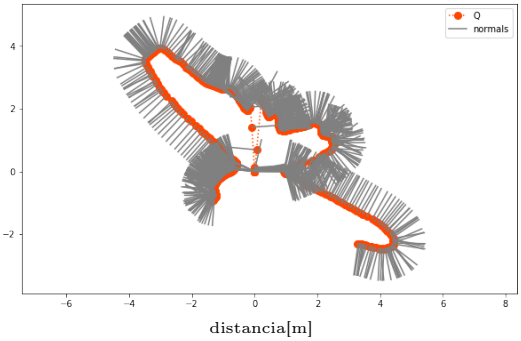


(a) Lectura de ejemplo para las figuras siguientes.

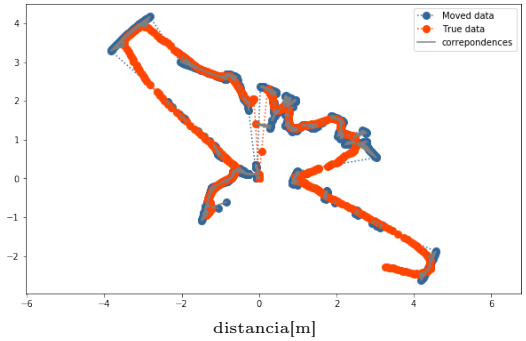


(b) Obtención de odometría láser comparando lecturas en muestras de tiempos cercanos.

Figura 5.18: **ICP** Entre observación en t y la observación t+1 para estimar odometría láser.



(a) Normales de los puntos de el símbolo que representa la lectura.



(b) Ejemplo de la lectura transformada encontrada por **ICP**

Figura 5.19: **ICP** Centroide y observación para estimar pose.

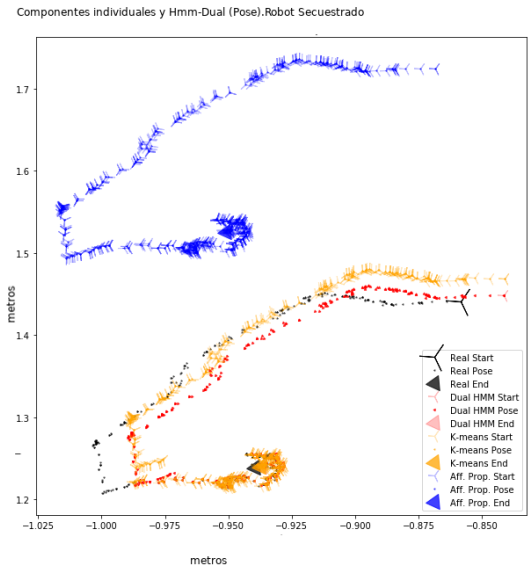
entrada del sistema, para corregir la odometría de la corrida de prueba, Como puede apreciarse en la figura 5.18b.

La eficiencia del modelo se incrementa si se cuenta con las normales de los centroides almacenadas previamente (ver figura 5.19a) y al utilizar **ICP** se obtiene la transformación rígida que las relaciona, y con ello, la odometría láser.

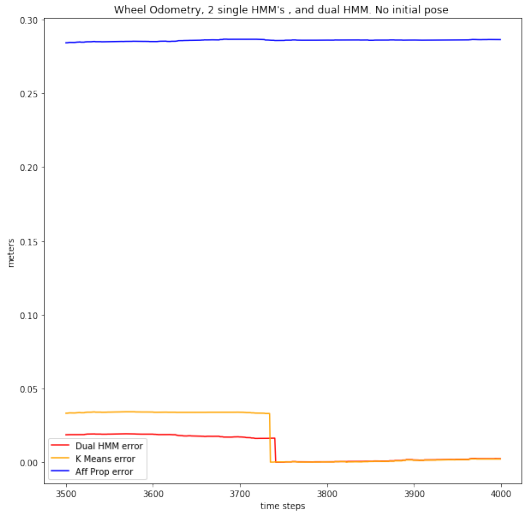
La figura 5.19b muestra la superposición de la lectura corregida, asimismo se reportan las métricas de error en un par de corridas de prueba realizadas en el laboratorio.

Finalmente se muestran las figuras 5.22a y 5.22b con sendas corridas de duración suficiente para apreciar el funcionamiento del sistema, ambas corridas se realizaron con el robot **HSR** en el laboratorio de Bio-Robótica. Es importante destacar que los recorridos abarcaron más de 25 metros lineales. De acuerdo con los resultados presentados en las figuras es dable concluir, ya que es evidente a simple vista, que las estimaciones de posición realizadas por el sistema se encuentran muy cercanas a las posiciones verdaderas.

Esta comparación se realzia de forma cualitativa, pues al no utilizar un mapa,

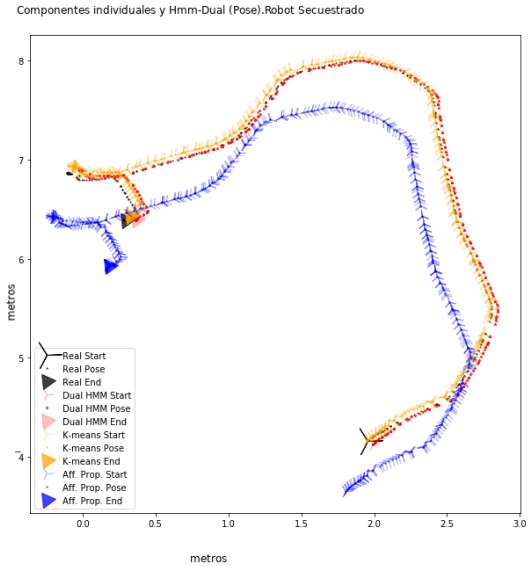


(a) Poses

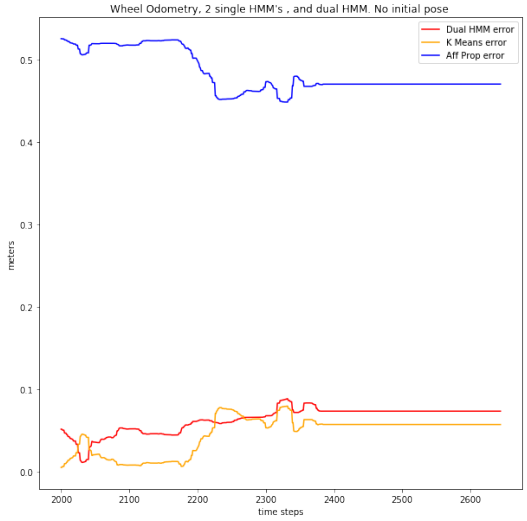


(b) Error absoluto de los estimadores mostrados en la figura 5.20a

Figura 5.20: Robot real, robot secuestrado

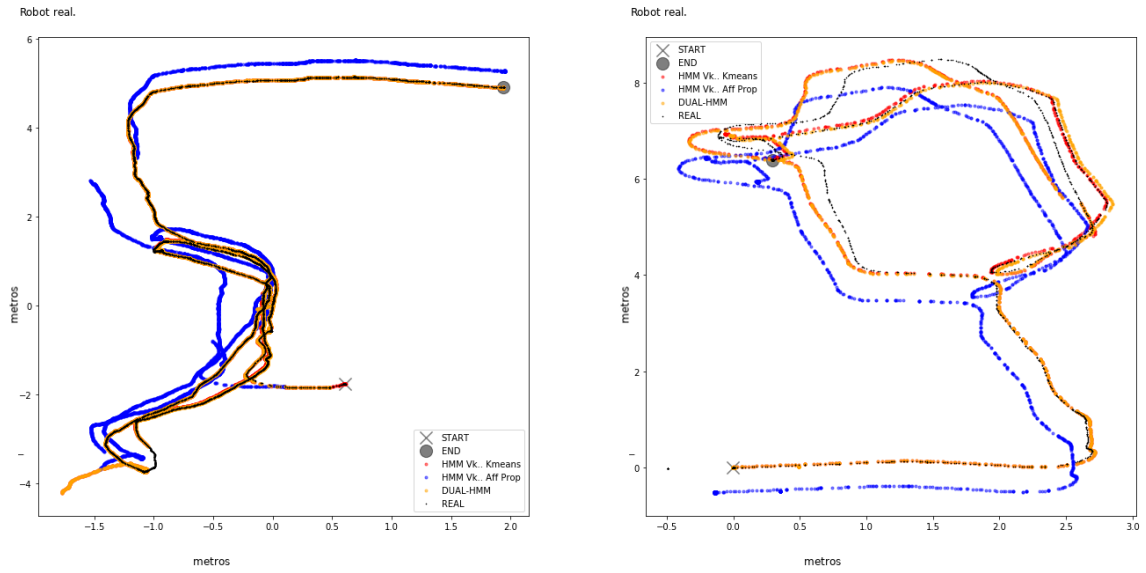


(a) poses



(b) Error absoluto de los estimadores mostrados en la figura 5.21a

Figura 5.21: Robot real, robot secuestrado



(a) (b)

Figura 5.22: Un par de corridas de larga duración con condiciones iniciales conocidas en el laboratorio de Biorobótica.

ni filtro de partículas, nuestro enfoque se basa sólo en la repetición. Es decir en la frecuencia con que se observan tuplos de estados y observaciones. En cierta forma nuestro método se basa en una sola partícula del algoritmo AMCL, primo cercano de los HMM's, miembro de la familia de algoritmos de localización de Markov 2.2.

5.4. Discusión

A lo largo de los experimentos descritos en este capítulo pudieron probarse muchas de las hipótesis y planteamientos teóricos que se investigaron. Pero más allá de un correcto funcionamiento, pudieron observarse comportamientos interesantes de la metodología propuesta. Como por ejemplo la modularidad, la capacidad de agregar nuevos nodos a un "mapa", así como la capacidad de entrenamiento a partir de muchos episodios de duración controlada, tal que permite confiar en la estimación de odometría de rueda.

Primero hubo que afinar algunos hiperparámetros de los modelos, como número de estados, velocidad máxima del agente, incluso el sensor(es) a utilizar.

Cómo se mencionó en [28] la tarea de evaluar un método de **SLAM** no puede ser tan trivial como un conjunto de corridas aleatorias. Es común encontrar que un modelo es más preciso en una u otra zona de el ambiente mapeado, incluso condiciones de luz o la presencia y número de obstáculos dinámicos son factores capaces de influenciar la decisión de si un modelo es adecuado o no.

Es por eso que en la sección 5 primero se evaluó a los modelos sólo en el espacio de estados valiéndonos de un *split* tradicional del conjunto de entrenamiento. Entonces

la primera evaluación de la suficiencia de modelar los modelos es la capacidad de localización. Una vez asegurada la capacidad de estimar, bajo el paradigma de robot secuestrado, la posición en el espacio estado, podemos evaluar la posición del agente cualitativamente al aplicar los métodos de odometría detallados en la sección 4.3.

Un sensor tipo LIDAR es relativamente poco útil para navegar pasillos, que aportan pocos *landmarks* con los que discriminar la posición. Por eso en la secc. 5.2 se detalla un esquema de fusión de LIDAR con visión que mejora el desempeño. Sin embargo, dada las tareas a resolver en las competencias citadas, esta habilidad no fue juzgada necesaria, dada la poca presencia de pasillos en la arena. Sin embargo, el modelo probó exitosamente el esquema de fusión de sensores de naturaleza diversa.

Por último a partir de la secc: 5.3.1 se muestran análisis cuantitativos de error de estimación en la posición. Donde una vez afinados los hiperparámetros se optó por una fusión de dos cuantizadores que actúan sobre la misma lectura de LIDAR.

La prueba última de funcionamiento del método , más allá de toda medición fue la capacidad del robot para navegar de forma autónoma en los ambientes reportados.

Es fundamental mencionar que el tiempo de respuesta no fue un factor Ya que el robot tarda más en llegar de un punto a otro que el algoritmo en realizar las estimaciones pertinentes, lo que garantiza su funcionamiento en línea.

6 Conclusiones

6.1. Conclusiones

Esta tesis reportó los resultados obtenidos para resolver el problema de **SLAM** mediante un sistema basado en Modelos Ocultos de Markov (**HMM**), la premisa consiste en obtener un modelo probabilístico del entorno explorado. Se utilizó un modelo estocástico para realizar las estimaciones de la posición del robot de servicio tanto en ambientes reales como simulados. Los resultados obtenidos durante la etapa de experimentación fueron alentadores y sugieren un solución versátil al problema de **SLAM**, basada en Modelos Ocultos de Markov **HMM** que puede ser implementada con varios sensores o con una fusión de los mismos.

Durante el desarrollo del proyecto se exploraron diversas fuentes de información, diversos tipos de sensores por ejemplo, Lidar Hokuyo, imagen 2D a color e información de 3D en forma de una nube de puntos (codificada con Octomaps [35]), etc.

La investigación se llevó acabo utilizando el robot **HSR** de Toyota que cuenta con todos los flujos información referidos, los resultados obtenidos fueron satisfactorios gracias al uso de un sistema de fusión de información ya que el sistema propuesto tuvo un desempeño sobresaliente en condiciones de poca luz o pocas características distintivas, para el sensor Lidar Hokuyo un pasillo de color provee poca información, un ambiente mal iluminado impacta de manera negativa en un sistema basado en imágenes, con ello se demuestra la ventaja en el uso de fusión de sensores.

En la industria existen sistemas capaces de resolver el problema de **SLAM** con niveles de optimización y desarrollo superiores. Sin embargo, los resultados obtenidos en el desarrollo del proyecto presentado en esta tesis sugieren que el sistema propuesto resuelve exitosamente el problema de **SLAM** con un costo computacional significativamente menor.

6.1.1. Características de interés.

Durante el desarrollo del proyecto se identificaron algunas características relevantes que vale la pena destacar:

- **Representación sin mapa.** Los resultados obtenidos demostraron que la interacción agente-entorno se encuentra completamente definida con los parámetros de los **HMM** y no necesita mapa de ocupación (ni de otro tipo), aunque

el sistema se puede beneficiar de un mapa para estimar la posición como en **AMCL**.

- **Sistema Modular.** La naturaleza del sistema es modular, es decir, se pueden agregar regiones nuevas sin necesidad de volver a entrenar todo el sistema.
- **Múltiples Sensores o fusión de sensores.** Los métodos de agrupamiento de sensores (fusión) permiten de realizar el proceso de cuantización vectorial.
- **Robot Secuestrado.** El sistema no depende del conocimiento previo de las condiciones iniciales, pero puede utilizarlas en caso de conocerlas.
- **Optimización en línea.** El sistema tiene la capacidad de ser optimizando en línea, lo que permite adaptaciones a pequeños cambios en el ambiente (p. ej. muebles que cambian de lugar) conforme ha realizado un mayor número de corridas exitosas.

El desempeño de una solución de **SLAM** es algo difícil de generalizar, y está relacionado directamente al tipo de ambiente, de agente, de aplicación, incluso de la ruta y de necesidades de precisión.

6.2. Recomendaciones para Trabajo Futuro

El proyecto reportado en esta teisis mostró como característica esencial la cuantización del entorno a un espacio de estados, lo cual facilita la adaptación e incorporación de algoritmos de aprendizaje por refuerzo que permitan navegar de forma óptima el grafo, o también permite incorporar algunas de las tareas propias de un robot de servicio como la manipulación y la interacción Humano Robot.

Se propone cómo área de oportunidad para trabajo futuro el desarrollo de la **navegación por campos potenciales**, si bien es una propuesta suficiente para la tarea de **SLAM**, la información con la que se dispone es suficiente como para hacer el planteamiento de una solución mucho más eficiente desde dicha perspectiva. Se propone un sistema de de **hipótesis de creencias**, que podrían ser comprobadas mientras el robot se mueve de un punto a otro de la ruta.

A Artículo

UDK 006.72

FUENTES OSCAR , SAVAGE JESUS , CONTRERAS LUIS

A SLAM SYSTEM BASED ON HIDDEN MARKOV MODELS

Fuentes, Savage, Contreras **A SLAM system based on Hidden Markov Models.**

Abstract. We present a graphe SLAM system based on Hidden Markov Models (HMM) where the sensor readings are represented with different symbols using a number of clustering techniques; then, the symbols are fused as a single prediction, to improve the accuracy rate, using a Dual HMM. Our system's versatility allows it to work with different types of sensors or fusion of sensors, and to implement, either active or passive, graphe SLAM. The Toyota HSR (Human Support Robot) robot was used to generate the data set in both real and simulated competition environments . We tested our system in the kidnapped robot problem by training a representation, improving it online, and, finally, solving the SLAM problem.

Keywords: localization, SLAM, robot navigation, mapping, Hidden Markov Model, sensor fusion, service robot

1. Introduction.

Service robots, such as the Toyota HSR [1], are increasingly becoming a part of our everyday life, so the ability to explore, map, and navigate its surroundings is of the utmost importance. SLAM, or simultaneous localization and mapping is a solution for this problem.

There are many accepted and well studied methods for solving SLAM, a brief overview of the main paradigms used for solving SLAM is presented on section 2. Depending on the application, one approach might be better suited than other. The sensors information available is also an important factor to decide which approach to use. e.g. Wheel information is an efficient way of estimating small changes in position, however a drone would not have this valuable information. Taking into consideration these differences may favour a SLAM method or type of sensor in a specific environment, or even a specific region in an environment. e.g., a dark corner may be a terrible place to use image-based methods, on the other hand, a colored flat wall would render little information to a LIDAR-based one.

We propose a graphe SLAM system based on HMM's (Hidden Markov Models). Our method can take advantage of several types of sensors measurements (or sensor fusion) while estimating a graphe topological representation. Dual-HMM allows us to simultaneously use two different quantizers, effectively fusing data at a symbol level. Furthermore, new nodes can be added without modifying the graphe (modularity), i.e. it can be grown or altered without having to train an entirely new model.

In summary, our main contributions are:

– A graphe SLAM system based on Hidden Markov Models.
– A modular system capable of using a wide variety of sensors and features.
– An autonomous training method.
– A navigation method capable of obstacle avoidance.
– A robust localization method.

The remaining of the paper is divided as follows. In Section 2 we present a summary of the SLAM problem and the HMM-based approaches. Then, in Section 3 we introduce our probabilistic approach to the graphe SLAM problem and in Section 5 we describe the experimental results. Finally, the main findings are discussed in Section 6.

2. Related Work.

The three main SLAM paradigms are Kalman Filter, Particle Filter and Pose graphe based implementations, and all of these can be found in the most commonly used open-source libraries.

2.1. EKF SLAM. The Extended Kalman Filter SLAM [2], [3] in one of the most accepted SLAM solution; it consists in three basic operations:

2.1.1. Robot Movement. The agent moves increasing its position uncertainty due to odometry errors.

$$S_t \longleftarrow f(s_{t-1}, u, n)$$

– f, motion model
– S_t, state of the robot at time t
– u, control signal
– n, noise

2.1.2. Discovery. The agent finds new interesting landmarks, which need to be referenced. The position uncertainty and sensor error readings are modeled using an inverse observation model i.e. where a landmark is in the map, given the scene as seen by the robot.

$$L_i = g(S_t, \vec{O}_t, y_i)$$

– g , Direct Observation model
– L_i, i-th landmark
– S_t, robot state at time t
– y_i, measure of i-th landmark

2.1.3. Re-Discovery. The agent finds a previously mapped landmark and re-estimates both its position and landmark position.

The extended Kalman filter has a “stage” for each of the above operations, making it a useful estimator for propagating the uncertainty related to the three mentioned actions.

$$y_i = h(S_t, \vec{O}_t, L_i)$$

– h , Indirect Observation model

The map representation itself is a matrix that stacks vectors of all mapped landmarks on any given robot state

$$map = \begin{bmatrix} S \\ L_1 \\ . \\ . \\ . \\ L_I \end{bmatrix}$$

– S , robot state

2.2. Particle Filtering. One of the most important characteristics of this SLAM approach is the building of an occupancy grid map. This map is later used to achieve localization with solutions similar to de EKF filtering, hence the name Filtering in Particle Filtering. Localization is achieved with Adaptative Monte Carlo Localization AMCL, a member of Markov localization algorithms. Murphy [4] et. al. proposed Rao-Blackewllized Particle Filters as a SLAM solution, the key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posteriori $P(\vec{x}_t, m \mid \vec{z}_t, \vec{u}_{t-1})$. The Rao-Blackwellized particle filter for SLAM makes use of the following factorization:

$$P(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = P(m \mid x_{1:t}, z_{1:t}) P(x_{1:t} \mid z_{1:t}, u_{1:-t-1})$$

Where x is the pose of the robot, m is the map, Z_t are the observations at time t, and U_t is the control signal.

Importance Sampling Filters (such as Sequential Importance Resampling or SIR) are employed to estimate the posterior $P(x_{1:t} \mid z_{1:t}, u_{1:-t-1})$, as mentioned by Grisetti et. al. in [4], where "each particle represents a potential trajectory of the robot. Furthermore, an individual map is associated with each sample. The maps are built from the observations, and the trajectory represented by the corresponding particle".

2.3. graphe SLAM. According to Grisseti et.al., "one intuitive way of formulating SLAM is to use a graphe whose nodes correspond to the poses of the robot at different points in time and whose edges represent constraints between the poses" [6].

From a probabilistic point of view, SLAM can be represented with a sequence of random variables as X (robot pose), M (map features), Y (sensor readings), and U (robot motion). To solve the SLAM problem, we simply use the maximum a posteriori probability:

$$P(X_t, M \mid Y_t, U_t, x_0)$$

for each time step t.

There has been a lot of work on solving this estimation problem [6] [7]. However, special attention has been given to [8], where the authors apply variable elimination techniques to reduce the dimensionality of the optimization problem. In the work presented in [9], a solution to active SLAM problem is proposed "in scenarios in which some prior information about the environment is available in the form of a topo-metric graph".

2.4. ROS SLAM. As mentioned before, many state-of-the-art solutions can be found as open-source libraries in popular frameworks such as the Robot Operating System (ROS). Here, we present some of the most popular implementations.

2.4.1. Gmapping. Gmapping [10] is based on Rao-Blackwelized Particle Filters as proposed by Grisetti et. al. in [11]. Particle Filters are a known application of Bayesian Filters in which a large number of importance weighted particles represent the a-posteriori probability; a probabilistic occupancy grid is used as a map representation and AMCL, Adaptative Montecarlo Localization, is used for localization in this map – AMCL is a member of the Markov localization algorithms family.

2.4.2. HECTOR SLAM. HECTOR SLAM does not use odometry information for the localization; it rather uses high update rates and low distance measurements to estimate the robot movement. A version of ICP is used to estimate the pose between samples and to maintain the robot pose estimate.

2.4.3. CARTOGRAPHER. A graph-SLAM approach proposed by the International's Karto Robotics in [12] – in Cartographer the nodes represent the pose of the robot and the edges the constrains between them. Nodes are usually defined w.r.t. contiguous nodes except when loop closures are detected where constrains for non contiguous nodes are introduced, which corrects the whole graph. Detecting loop closure is not trivial; in the ROS

implementation, scan matching is performed by Sparse Pose Adjustment (SPA) [13]. Cartographer uses an occupancy map in order to estimate position where the map representation is done via Gmapping.

As we can see from this brief review, some methods are better suited for specific tasks or robot architectures. Our research is based on the idea that each method has its strengths and we propose a method, that, focusing on versatility of application, tries to take some of the advantages inherent to these methods.

3. HMM-based graph-SLAM.

We propose a versatile graph-slam system based on Hidden Markov Models; the goal is to estimate a topographic graphe given noisy sensor measurements and pose estimates. In this section we will briefly describe the core concepts in our implementation.

3.1. Hidden Markov Models. A Hidden Markov Model (HMM) is a two random variable stochastic process, in which only one of the random variables is directly observable. In its discrete version, and provided the Markov property [14] is fulfilled, the system dynamics is fully defined by a transition matrix A, an emission matrix B, and, optionally, an initial conditions vector $\vec{\pi}$.

$$a_{ij} = P(S_t = s_j \mid S_{t-1} = s_i)$$

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1N} \\ a_{21} & a_{22} & ... & a_{2N} \\ a_{31} & a_{32} & ... & a_{3N} \\ . & . & . & . \\ a_{N1} & a_{N1} & ... & a_{NN} \end{bmatrix}$$

$$b_j(k) = P(V_k \mid q_t = S_j)$$

$$B = \begin{bmatrix} b_{11} & b_{12} & ... & b_{1M} \\ b_{21} & b_{22} & ... & b_{2M} \\ b_{31} & b_{32} & ... & b_{3M} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ b_{N1} & b_{N1} & ... & b_{NM} \end{bmatrix}$$

$$\lambda = [A, B, \vec{\pi}]$$

We propose the hidden random variable to represent the pose of the robot, and the observable variable will be a discrete representation (quantiza-

tion) of the sensor measurements. The dynamic programming based algorithms' family related to HMM's [15] provide efficient estimations to probabilities related to the localization problem.

Our approach on Markov localization uses a discrete code-book to represent the sensor's readings. The symbolic representation of the readings allows for any sensor, or fusion of sensors to be used as observations. Localization is achieved, as any other Markov localization approach, by estimating the probability of being on state $S_t = s_n$ given a last known state $S_{t-1} = s_n$, a sensor(s) observation symbol $O_t = V_k$, and a control signal $U(t)$. The 2D pose state space is defined by the tuple $\langle x, y, \theta \rangle$.

For each time step t, random variable S can take the value of a discrete set of N states; similarly, the observations of each time step will be a member of the discrete code-book observation set of K symbols V. Control signal $U(t)$ is assumed constant, and only an on-off signal for the potential fields autonomous navigation system is necessary, as follows:

$$S_{t-1} = s_n, O_t = V_k, U(t) = u$$

An action set is used for the active slam pose belief exploration. Planing path is then obtained using a search algorithm like Dijkstra or A*. Pose traces are proposed so given a state belief and an action the most likely future state is estimated.

The algorithms typically associated with HMM's can efficiently estimate this probability using dynamic programming. Even tough number of states in the models is reported as fixed, it was done to have uniformity between models, since models are modular and can easily be connected to form a global model, similar to sub-maps in Cartograher. The balance between computer costs and accuracy can be fine tuned with the number of states, and the observations code-book size.

The resulting representation of the environment is a symbolic pose graph, where each node or state is the centroid of the free region surrounding it; orientation is also accounted for with a scaling factor. The HMM model is completed by recording in the code-book all the observations sensed while the agent is at any given state. As the agent acquires more information by exploration, new states are formed or the existing ones moved to account for new data, eventually including all the explored area. The number of total states is a hyperparamter of the model, and it can be seen as a scale factor to be used. The more states in which free space is divided, the more accurate the correction, at the expense of more computational expenses.

One of the main advantages of the method is adaptability, since it can use any sensor or fusion of sensors. As briefly mentioned above, observations are not used directly. A symbol, part of a discrete data set, is used to represent sensor readings. This quantization of readings allow the method to be used with various sensors, different sensor coding, or even sensor fusion. In HMM terms, we have various emission matrices for each transition matrix. We perform estimation on various models, each related to each sensor, or fusion of sensors. The model likelihood given a set of observations is easy to obtain with HMM algorithms like Forward Algorithm, (eq. 2) and it's Backward counterpart,(eq. 3), and it's used as a metric to decide which model to use on a specific area. A lidar will gather more information from a dark corner, while a camera would do better on a long hallway with distinctive visual landmarks. Computer cost is in the order of N^2T (eq. 4), however, the number of states rapidly increases the computer cost, "the curse of dimensionality", where the direct calculations without dynamic programming algorithms is in the order of N^T (eq. 1).

$$P(\vec{O} \mid \lambda) = \sum_{j=1}^{S} \left(\pi_{s1} \prod_{t=1}^{i=T} a_{s_t . s_{t+1}} b_{s_t - s_{t+1}}(o_i) \right) \tag{1}$$

$$\alpha_{t+1}(i) = \sum_{i=1}^{N} (\alpha_t(i) a_{i,j})\, b_j(o_{t+1}). \tag{2}$$

$$\beta_t(i) = \sum_{i=1}^{N} (a_{i,j} b_j(o_{t+1}) \beta_{t+1}(j)\,. \tag{3}$$

$$P(\vec{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(i) \tag{4}$$

Finally, localization is performed using Viterbi algorithm; this algorithm estimates the most probable sequence of states given a sequence of observation symbols. The model with maximum likelihood is obtained with the Forward Algorithm. The most likely model given the readings is used in the Viterbi algorithm, with $max\left[P(\vec{S} \mid \lambda, \vec{O})\right]$ being the discrete representation of the space in which each state is the closest centroid to the robot state at the time of sensor capture. The size of the free region represented by each state is variable, but it's kept to a small enough area as to make wheel odometry reliable. The transitions between states happen on a well defined regions normally distributed around a transition point. The most frequent transition point between states is used as a "reset" wheel odometry w.r.t. that point. Similarly to cartographer's sub-maps, wheel odometry is reliable in small

regions.

4. Implementation.

We have presented in a very general way all the components of the system we propose. Now, we will show some interesting implementations using those components. It is important to mention that a map is not necessary in our method but, in case one is available or needed, given that our method deals with the same probabilities as the particle filter used in AMCL [16], it is possible to use Rao-Blackwellization. Furthermore, as our training methods create a decent grid estimation, whether using a map or not is greatly dependant on the application and error scale needed.

In our implementation, a global reference frame is created, however, global mapping based on iterative closest point (ICP) methods is also a possibility we explored in some regions, specially when a high accuracy in localization is needed.

In our SLAM representation, we have wheel odometry estimate w.r.t. to initial position of the robot, a Hokuyo 2D Lidar scan for observations, and a signal control and a world representation is estimated. i.e. a HMM Model is trained as follows.

4.1. Training. Short exploration runs are conducted, gradually restarting to a known value, like an entrance or a recharging station;this episodic learning approach makes the wheel odometry reliable. i.e. the duration of the episodes is such that wheel odometry is reliable. The control signal in exploration mode is an on/off signal enabling a reactive potential fields behavior [17]. Some additional constraints are added to the behavior, as mentioned in our previous work [18], like a exploratory turn every given time samples, or an artificial attraction towards doors.

Model training is done with a labeled training set. This training set is formed by a vector $\vec{O}_t$ that contains all the 720 laser readings from a laser sensor and a odometry-based pose vector or hidden state S_k.

Baum-Welch [15] is the most commonly used method for the HMM model estimation. It is an elegant approach that calculates the optimal model parameters given the observations, it is also an Expectation-Maximization (EM) algorithm [19]. The optimization is done by maximizing the Likelihood of the Model given some readings. The most common training method is used offline and will serve as an initial estimation for our method, i.e. we maximize the likelihood of the model given some observations (training set). An auxiliary variable ξ is defined. This variable can be easily represented with the Backward and Forward variables, as follows

$$\xi_t(i,j) = P(s_t = i, s_{t+1} = j | o_t, \lambda)$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{T+1}) \beta_{t+1}(j)}{P(o|\lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{T+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(o_{T+1}) \beta_{t+1}(j)}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

$$\sum_{t=1}^{T-1} \gamma_t(i) = S_i^T$$

Where S_i^T is the number of transitions from S_i over T time steps.

$$\sum_{t=1}^{T-1} \xi_t(i,j) = S_{ij}^T$$

Where S_{ij}^T is the number of transitions from S_i to S_j over T time steps.

$$\hat{a_{i,j}} = \frac{S_{ij}^T}{S_i^t}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)}$$

$$\hat{b_i(k)} = \frac{\sum_{t=1}^{T} 1_{o_t = v_k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

[15] uses $\sum_{t=1}^{T} 1_{o_t = v_k}$ to represent number of times V_k is seen in state j.

We also use a gradient descent implementation of the Baum-Welch algorithm [20] to keep optimizing the model online. This can introduce subtle changes when the environment changes, or when a new area is mapped and added to the global model. A mini-batch approach can also be used to re-estimate the model every time a batch buffer is filled. Each new reading contributes to an infinitesimal change in its respective row and column in the transition matrix. Note from eqs. 5 and 6 that change only happen on the row and column related to the reading. Least frequent poses will be absorbed by

the more frequent (desired) readings. This online process allows aligning the new readings and reinforcing into the matrix the optimal values given a space and obstacles configuration.

$$\frac{\partial P}{\partial a_{ij}} = \sum_{t=1}^{T} \alpha_t(i) b_j(O_{t+1}) \beta_{t+1}(j) \tag{5}$$

$$\frac{\partial P}{\partial b_j(O_t)} = \begin{cases} \delta_{j1}\beta_1(j) & t = 1 \\ \sum_{i=1}^{N} \alpha_{t-1}(i) b_{ij}(O_{t+1}) \beta_t(j) & t \neq 1 \end{cases} \tag{6}$$

Due to the low complexity for estimating a model and, later on, navigating the model to obtain information, it is possible to use several models simultaneously; this allows us to add new nodes (i.e. states) to the model without the need to re-estimate the whole model.

In case that autonomous mapping is not required, and human interaction is used to "explore" the scene, the model will benefit from different strategies, and an action policy estimate can be obtained. Again, versatility is an important characteristic of our method. Regardless how the training set is obtained and the model estimated, that is, either offline, online , batches, etc. an HMM is obtained with the Baum-Welch algorithm. Figure 1 shows a topological node graphe obtained from the transition matrix of the proposed HMM after online navigation. Results found on Section 5 and in Fig. 10 and Fig. 11 show this process in a standard competition arena where an apartment is explored, and an initial graphe estimated; the green arrows represent the state <x,y,θ>. Background image is just a reference for illustrative purposes, since map is not being used.

4.2. Observations. Observations used for the estimation of the HMM are quantized (and finite) in nature, so the method used for the quantization or the kind of the sensor itself is not important as long as we use a finite code-book to represent them. We explored various observation alphabets for the code-book where Lidar readings were treated using different clustering techniques, namely K-means and affinity propagation (Figure 2). Some experiments were also made by fusing LIDAR and 2D images features obtained with Resnet [21] architecture.

The final pose estimator efficiency greatly depends on good readings. "A good reading is that which can be re observed, and one easily differentiates from other reads" [22].

4.2.1. Lidar. K-means (or mini K-means in this case) works good when a large amount of data is used. The low occurrence of outliers takes them out of the estimation; however, highly repeated values will "skew" the means

resulting in some information loss. A different algorithm is proposed (Affinity Propagation) [23] to quantize the laser readings based on their similarity, rather than their frequency of appearance. A dual HMM is introduced, which estimates independently using code-books from each clustering algorithm, and trusting only matching estimates obtained independently by each model. A block diagram of the proposed DUAL HMM architecture can be seen in

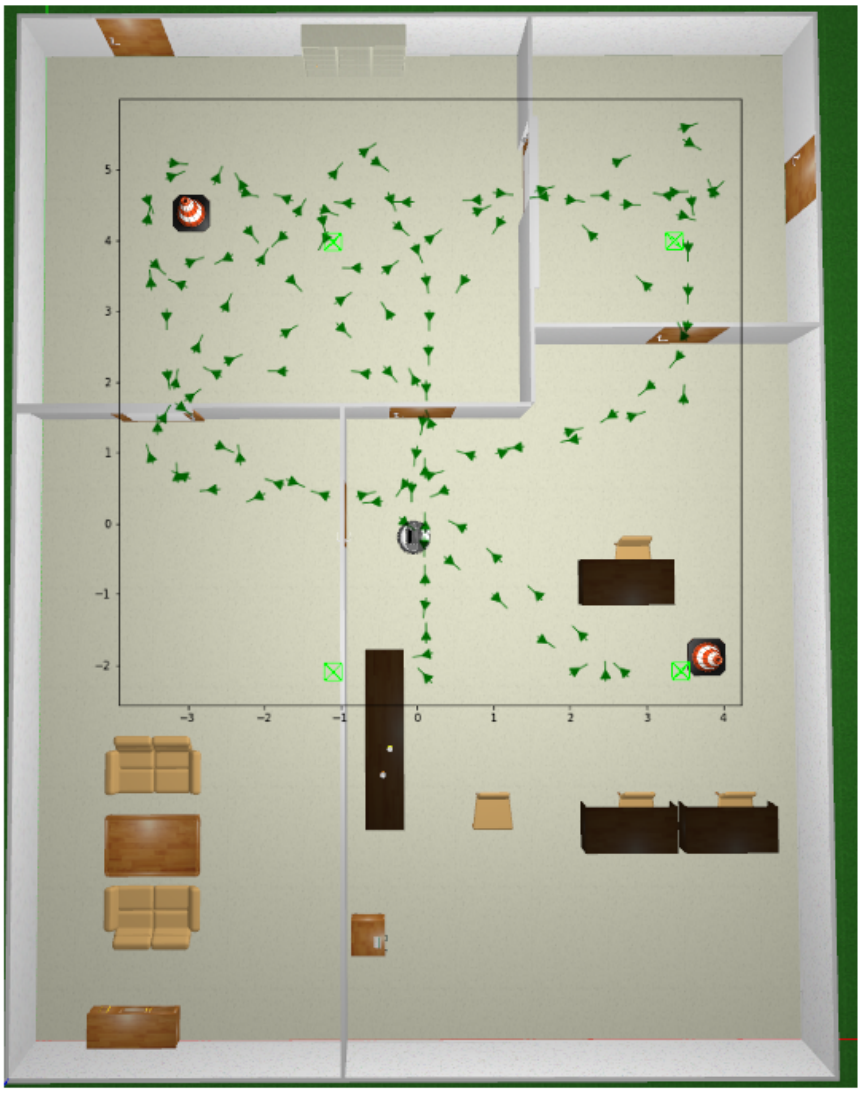


Fig. 1. Topographical graphe representation, obtained from the transition matrix of the HMM trained. Graphe is created with the states corpus (2D poses) shown as green arrows.

figure 3 The statistical characteristics of most readings are quite similar and this similarity enables the usage of affinity propagation clustering techniques to generate more heterogeneous clusters; the drawback, however, is the lack of scalability when there are many samples. K-means, on the other hand, works well with large data sets and, furthermore, being a Euclidean distance approach, it can be used online (with the previous centroids).

In sum, we use measurement symbols as anchor of information and depending on the sensor's nature, on the environment conditions, and even on the route, one kind of symbol generation might result with a better performance. A robust SLAM system should benefit from all symbol representations; Sensor Fusion [24] is used in this way, yet we propose a much simpler approach: Dual-HMM. In our experiments, we use two different features extracted from the same observations vector, but this applies to any kind of observation.

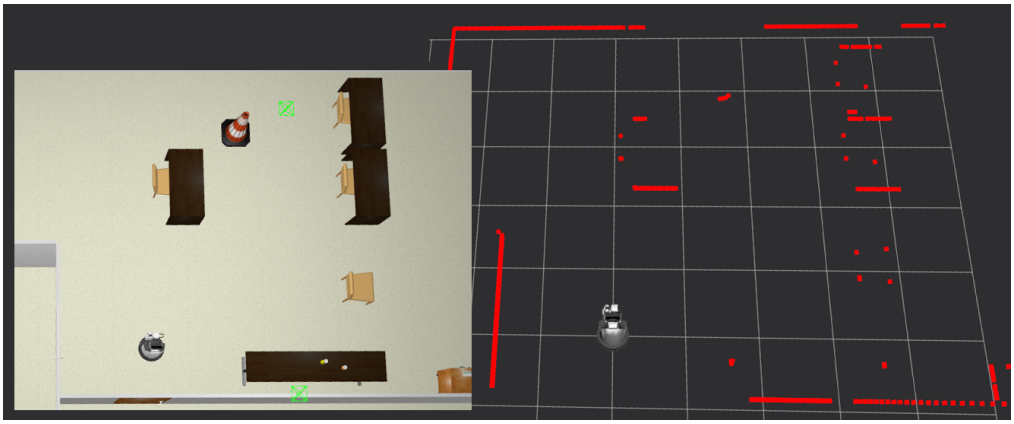


Fig. 2. Apartment scene with its corresponding Hokuyo Lidar reading.

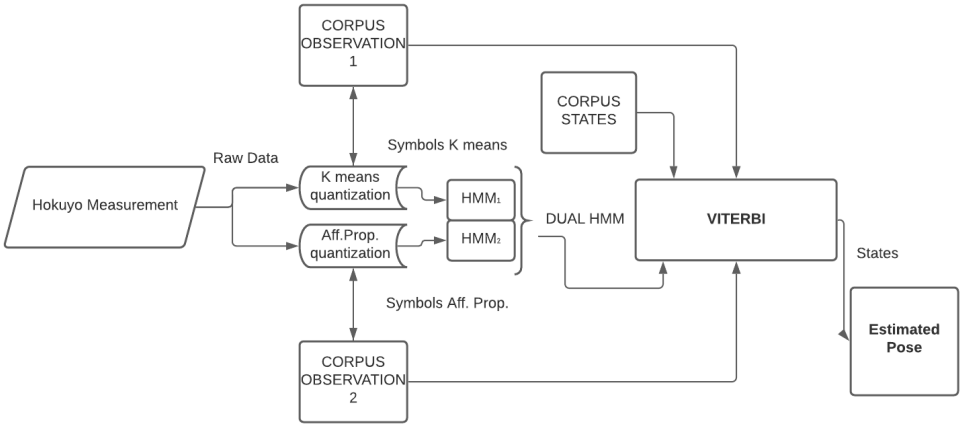


Fig. 3. Block diagram of the proposed DUAL-HMM system

Fig. 4 shows a common scene of a service robot, and on the right side an example of a typical lidar measurement in green; the centroid obtained with K means in orange and the Affinity Propagation exemplar in green.

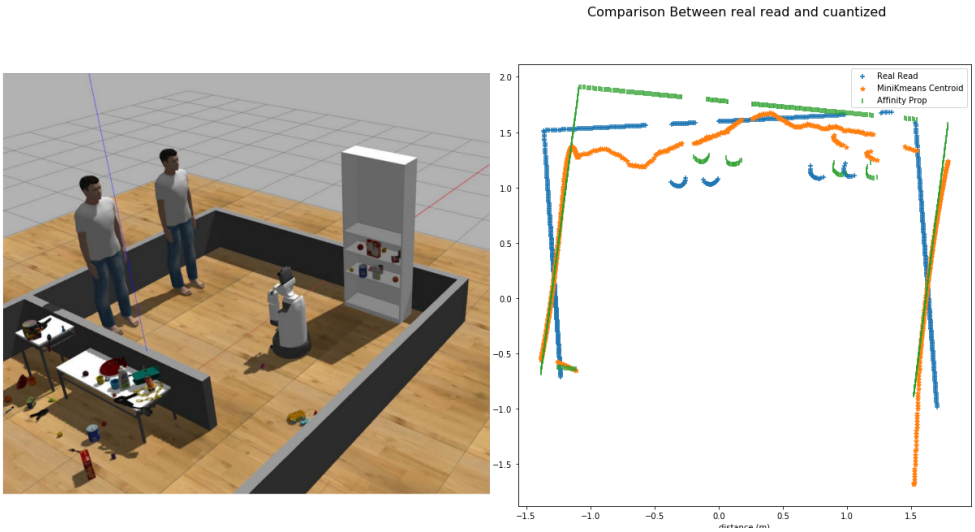


Fig. 4. Example of a Hokuyo typical reading (in blue) with its K-means centroid (shown in orange), and affinity propagation exemplar (green color).

The final pose estimator's efficiency greatly depends on good readings "a good reading is that which can be re-observed, and one easily differentiates from other readings" [22]. K-means (or mini K-mean, in this case) works well when a large amount of data is used since the low occurrence of outliers takes them out of the estimation; however, highly repeated values will skew the means. Affinity Propagation [23] is another clustering algorithm and it was used to quantize the Hokuyo readings based on their similarity, rather than their frequency of appearance. Fig. 5 shows K-means information loss in the most observed symbol. In comparison, Fig. 6 shows the symbols assigned by the more computationally expensive Affinity Propagation to a series of new readings.

In our approach, a dual HMM makes independent estimates using symbols from each clustering algorithm. Although there are other methods to achieve sensor fusion, in our proposal, we use a simple symbol level data fusion.

4.2.2. Resnet feature extraction. Another observations alphabet was quantized by getting an observation vector $\vec{O}_t$ from the last layer of a Resnet 50 CNN (Convolutional Neural Network). Such vectors were quantized into an alphabet and its respective emission matrix estimated. All models share the

same transition matrix. Results reported on section 5 show models using the fusion of a Resnet symbol with a lidar K-means symbol.

4.3. Localization. Once a model, or several models, are found, it is possible to estimate the robot's pose with a set of past observation symbols $\vec{O} = V_t, V_{t-1}, ..., V_{t-M}$. The most direct approach is to the Viterbi algorithm [25] that takes a sequence of quantized observations $\vec{O}$, an HMM model λ, and

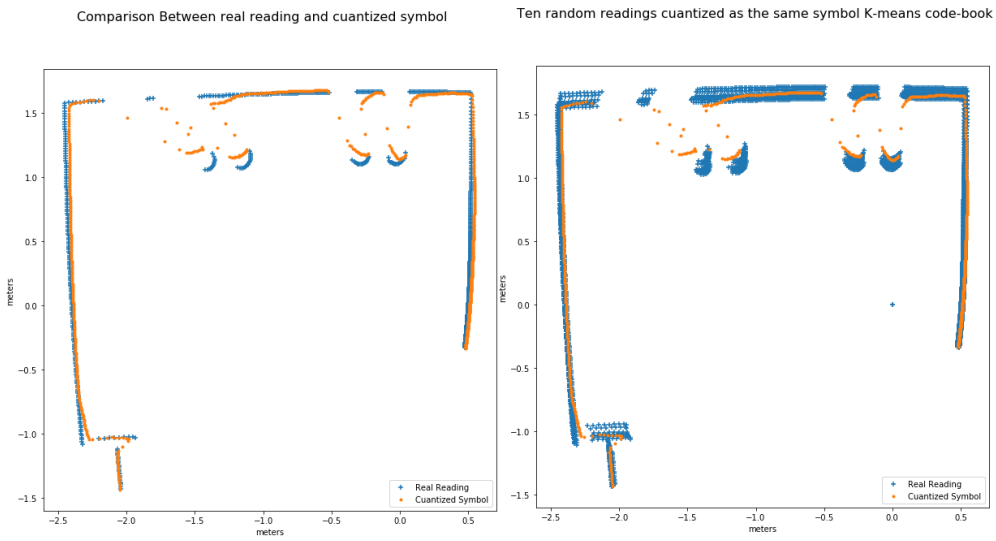


Fig. 5. Most common read and 10 samples clustered in the same K-means alphabet symbol

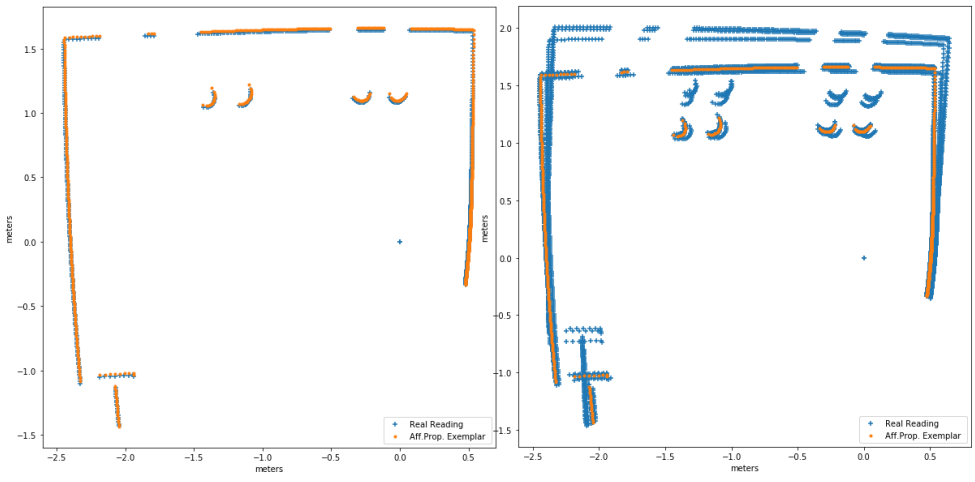


Fig. 6. Most common read and 10 samples clustered in the same affinity propagation alphabet symbol.

an initial conditions vector $\vec{\pi}$, and yields the most probable state sequence traversed by the system given the observations and HMM model.

$$P_1 = \pi_i \cdot b_i(o_1)$$

$$P_t(j) = \max_{1<i<N} [P_{t-1}(i) \cdot a_{ij}] \cdot b_j(o_T).$$

$$S_t(j) = arg \max_{1<i<N} [P_{t-1}(i) \cdot a_{ij}]$$

$$P\left(\vec{S} \mid \vec{O}, \lambda\right)$$

The length M of this observations vector is called the Viterbi buffer, and it is a hyperparameter that should be trained to optimize the models. Forward and Backward algorithms are used to find the most common state sequence and the probabilities of being in a specific state given the observations.

4.4. Pose Correction. Wheel odometry is an easy way to estimate a pose with the information from the wheels and the control signals. This estimation is reliable in a short term because a small error – due to slips and control noise – is accumulated over time, i.e. a reference point needs to be maintained. The accumulated error makes the estimation unreliable in the long run; this error is normally distributed [26] so, it is possible to characterize a particular floor-wheel interaction odometry error.

In our proposal, wheel odometry is corrected by resetting the reference point where the correction is made when a state transition is detected via the Viterbi algorithm. It is easy to observe from the training set that the transitions from one state to another usually happen approximately in the same pose value, i.e. transitions happen in a normally distributed point with the same distribution as the wheel odometry error; this value is used as the correction value on trusted transitions. Wheel odometry can be trusted inside the small region related to a specific state or centroid or in a time interval.

Fig. 7 shows in green the position estimate of the Wheel odometry in a run traversing through two states; black points show the real pose of the robot. Two independent HMM's correct the position according to their own estimates – it can be seen in the upper part both models estimate the same pose, and correction is the same.

4.5. Navigation. After the training process, the possible states of the system are represented with the pose centroids corpus; the topological map used by the Dijkstra algorithm is a version of the HMM's transition matrix. We propose different kinds of nodes; some of them are bidirectional, others

one-directional, some others only landmarks used to re-estimate the position. This is called *labeling* in some literature relating to HMM and must not be confused with the labeling of the training set. Again, the graphe nodes (or pose centroids) are re-estimated online with each reading and it tend to align with the optimal route as the robot navigates by being attracted to a single centroid at a time (virtual attractor). Dijkstra route will be then a path containing the sequence of centroids to visit to get from a current position to a goal. Potential fields reactive behavior is used to sequentially visit these virtual attractors while avoiding unmapped or dynamic obstacles. Figure 8 shows a sequence of images of the robot traversing a path made of such attractors – the control signal is simply an on/off signal, and the robot is either trying to get to the next attractor or it is static on a fixed position.

5. Experiments.

We present results using a Gazebo simulated home environment, a typical indoor setting for service robots [27]; even though a map is provided with the simulated environments, we do not use it in our SLAM tests, except for figure clarity.

5.1. Benchmark. We use a benchmarking method proposed in [32] to compare our models. Dieter et. al. propose "a metric for measuring the performance of a SLAM algorithm not by comparing the map itself but by considering the poses of the robot during data acquisition". As mentioned, a

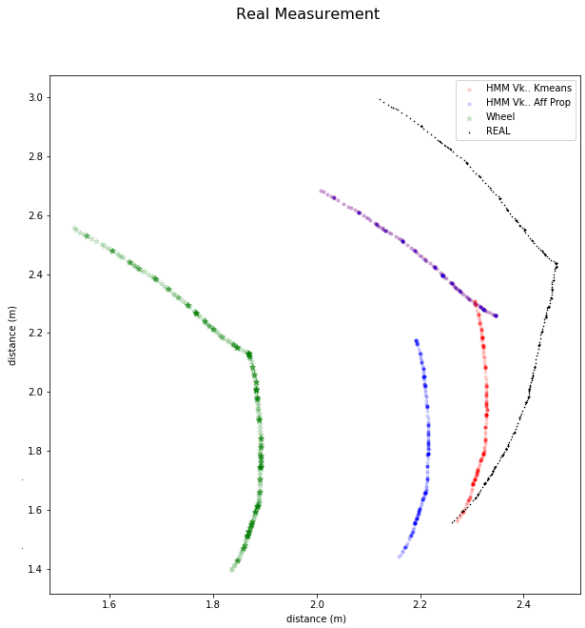


Fig. 7. Real pose in black and three different pose estimation methods. Wheel odometry in green, a single HMM(K-means) in blue and single HMM(aff.prop.) in blue.

map might be useful for human interaction, and feature comparison, but there are many other representations that can work as a "map" and, although our method could be used to build an occupancy grid with gmapping, the map itself is not required for our SLAM method, and is only presented for clarity.

The metric proposed also allows to compare methods that do use a map with others that do not, since it only relies on estimates of the trajectory of the robot given by a set of poses where the observations are taken.

5.2. Training. First, the robot roams the environment reactively and, at the same time, registers wheel odometry and laser measurements at every time step. In order to make wheel odometry reliable, episodes are run among known fixed locations like doors or furniture, or in short periods of time – this serves as ground truth comparisons for the localization task and odometry correction metric.

Once enough training data is obtained (around 15k samples in our experiments), two HMM's are calculated, both HMM's share the same transition matrix and, as a consequence, the same topological representation; each HMM, however, uses a different code-book to represent the observations (K-Means and affinity propagation, respectively). Each code-book favors different areas of the world so, dual representation or sensor fusion can be applied; however, we opted for a dual HMM. The dual approach ensures that only correct estimates are used for correction; as a consequence, wheel odometry is corrected only when both HMM's yield the same estimation given the observations.

The pose can come either from regular uncorrected wheel odometry or, ideally, corrected odometry. In the simulations, Gaussian noise with the parameters obtained from the real robot was added to the ideal training set as a data augmentation pre-processing technique.

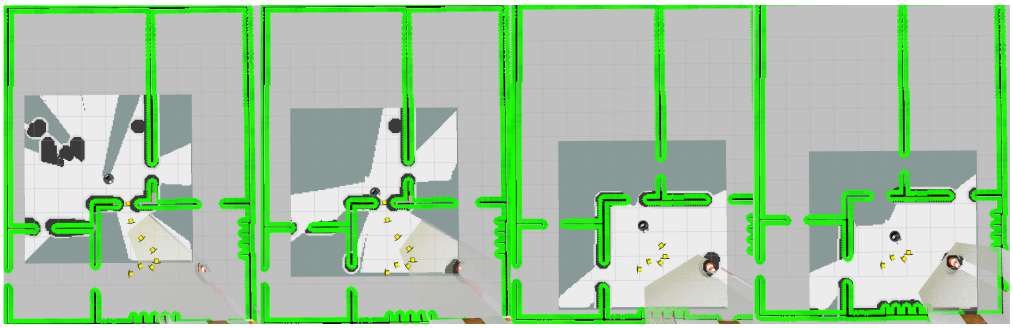


Fig. 8. Different moments of reactive navigation through a route formed by virtual attractors. Route is produced after planning using Dijkstra to navigate the joint transition matrix of the Dual-HMM.

Additionally, to test the results of the different models we later perform around a 100 sample runs in the arena with no initial information and random routes.

In Fig. 9, an example run is shown. Pink points represent older real poses (still unreliable since no initial conditions are used for estimation) and yellow points are poses with a big enough buffer to be reliable. The Figure shows an example of a correct estimation, this problem is commonly known as Kidnapped Robot [30], meaning no initial conditions information is used for the estimates; however, once a correct estimation is made initial state probabilities are available for further estimates, greatly improving accuracy. Green dots represent the poses centroids, i.e. the corpus of the HMM hidden variable X, and blue and red dots represent real and estimated hidden states, respectively. Wheel odometry correction is not applied at this stage, and accuracy is tested with the quantized states, not the real odometry, i.e. correct estimations on the HMM's with no initial conditions $\vec{\pi}$.

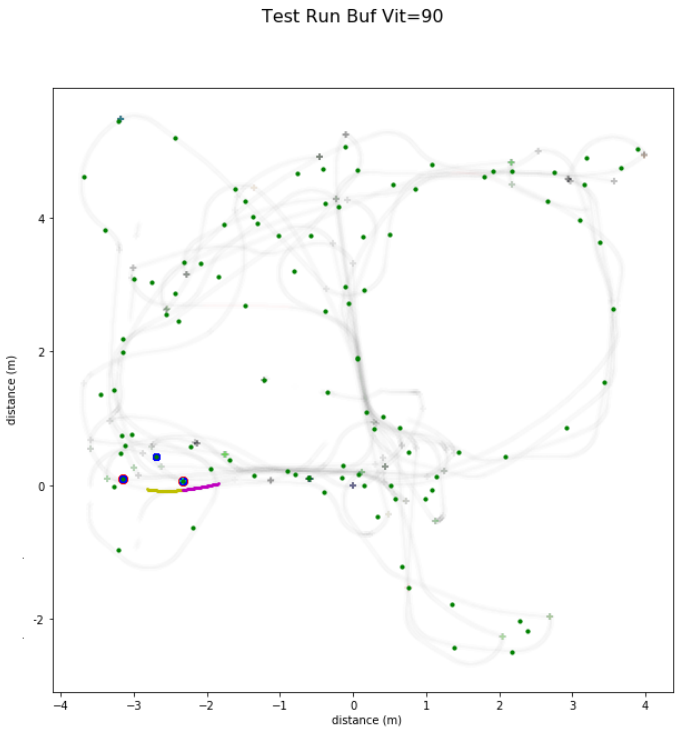


Fig. 9. An example run with a correct estimate and no initial conditions information traversing three states. State corpus in green, estimates on red and blue, beginning of the test run in pink.

5.3. DUAL-HMM. The simple HMM using K-means Lidar alphabet was enough for the WRS tasks but, in order to test various data fusion schemes,

a dual HMM is trained with the same training set – dual HMM uses both Lidar alphabets where the K-means and the Aff. Prop. symbols create each an emission matrix for the same transition matrix. Finally, a fused data model is proposed where the alphabet code-book is the fusion of 2D Lidar readings with the normalized features obtained from the Resnet CNN. Even though some information can be obtained this way, it is highly correlated with the random starting point because, by nature, there are regions with better features than others and, even using initial information, no significant improvement to the model is found.

In our experimental setup, the considerable computer expenses related to CNN's do not make it worth it to solve the robot relocalization problem, since accuracy does not improve significantly. This fusion may favour a different application, robot configuration or environment, and, since we show data fusion using different code-books, it is included in this work as a possible configuration for visual SLAM.

5.4. Modularity and Online Improvement. To show modularity, we use a multiple room environment. New graphs (or states) can be added after an initial HMM was found, or even a completely new HMM for each room; there is no need to retrain the whole model, as our previous research suggests [18]. Figs. 10 and 11 show the centroids aligning closer once enough online training had taken place. Online training yields a more structured and organized topological representation of the explored environment (shown in Fig.11), i.e. the topological representation keeps improving as the robot successfully navigates it thanks to on-line Baum Welch algorithm. Figs.10 and 11 show the topological graphe representation before and after online improvement.

5.5. Pose Correction . A different virtual environment was used to validate wheel odometry correction. The reality gap is broken since the training data obtained from the model was used to navigate the real world arena, both used in the World Robot Summit 2018 and 2020 (WRS) competitions. Figs. 12 and 13 show the arena and the topological graphe found.

Again, the robot was trained by acquiring the training set on a potential fields autonomous exploration run. Since the dimensions of the arena made wheel odometry reliable enough, the whole arena was modeled in the same HMM transition matrix (Fig. 13).

The test runs shown do not use initial state information (kidnapped robot), however, this initial information is available after a good estimate, and could be used for maintaining the agent's pose estimate. Such case is shown in fig. 14 where a comparison between the real trajectory and the components (K-means in orange and Affinity Propagation in blue) of dual HMM, as well as dual HMM in red; the real trajectory is displayed in black.

In case that a good enough current initial position is known, wheel odometry is a very good way to estimate pose, once error starts growing it can be seen that Dual HMM keeps pose estimates better aligned on the long run. It is important to note that HMM has a small quantization error on the first estimation because there is no initial information, and as such no initial

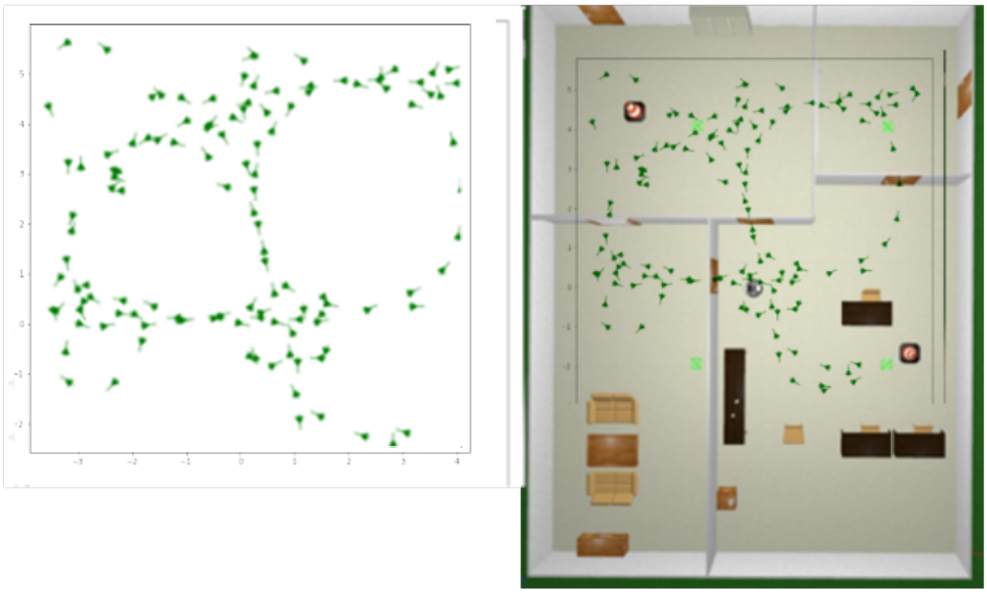


Fig. 10. Initial topological representation of the environment as obtained from off-line training.

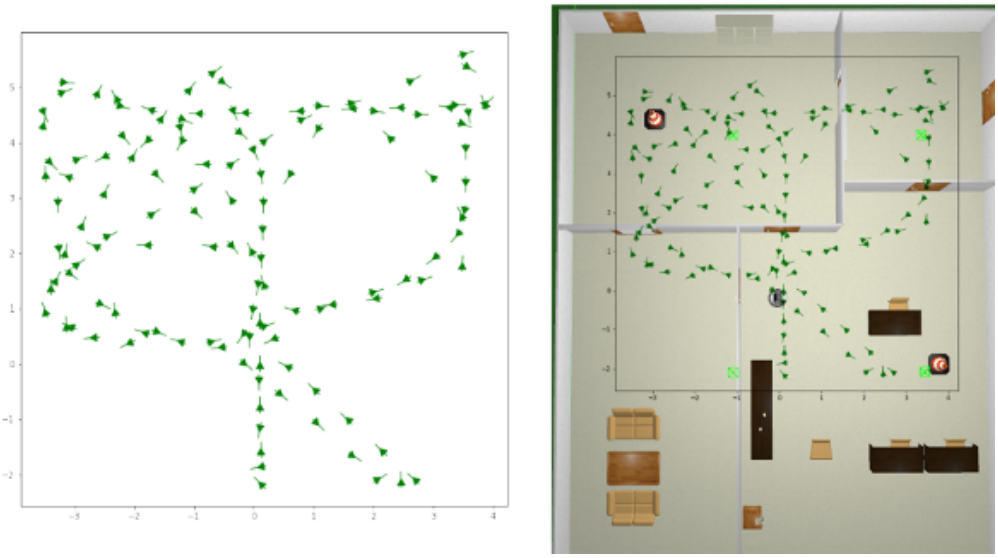


Fig. 11. A more structured and organized topological representation of the explored environment, thanks to on-line training.

Fig. 12. World Robot Summit arena with the proposed topological graphe representation.

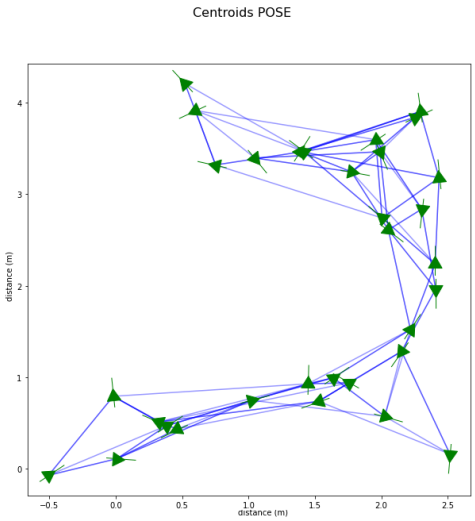


Fig. 13. WRS arena HMM representation Topological Map graphe.

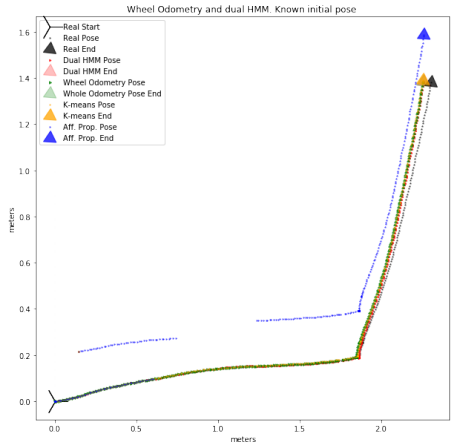


Fig. 14. Comparison with known initial location of Wheel Odometry estimate with known initial conditions in green , Dual-HMM pose estimate in red. Each of the dual single components observations K means in orange and Affinity Propagation in blue. Real trajectory shown in black.

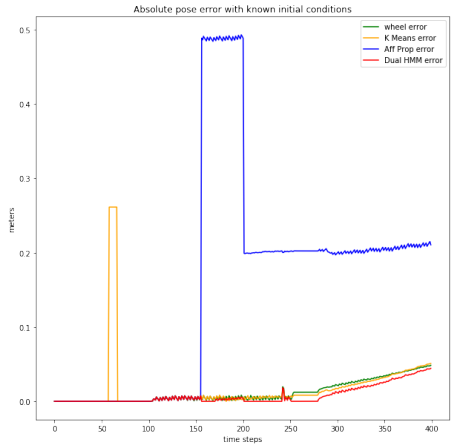


Fig. 15. Absolute pose error of fig. 14

transition, just the centroid of the hidden state is available in the first kidnapped robot esstimate. A much longer run is shown in fig. 16 to further illustrate this idea. Figs. 15 and 17 report the absolute error in pose estimate over time.

5.5.1. Kidnapped robot. The wheel odometry has no initial information, and no way to re-estimate its real position, so the error keeps growing as time goes on, unlike the dual HMM odometry; as the agent continues to move it keeps traversing reliable states and correcting its pose. Results shown in Fig. 18 and fig. 19 suggest that the kidnapped robot problem was solved once enough time to traverse a trusted transition had elapsed; however, in case

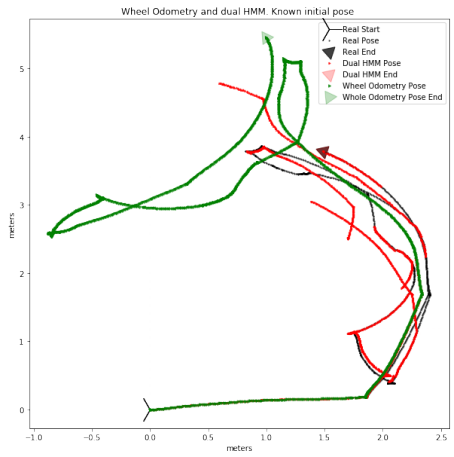


Fig. 16. Same initial conditions than figure 14 but a much longer run. Wheel odometry in green clearly diverges while Dual-HMM in red remains operational trough the whole run.

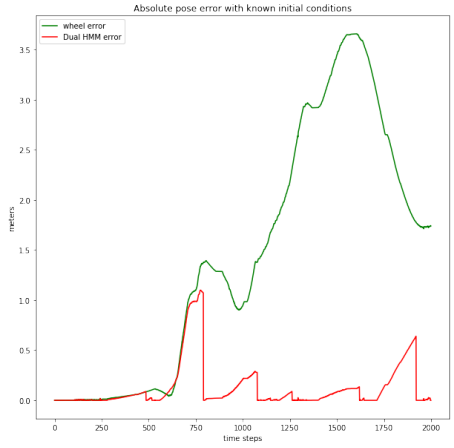


Fig. 17. Absolute pose error of a long run.

wheel odometry pose has initial conditions, it is a better estimate for the first few time steps.

5.5.2. Catastrophic Event. An example run before such event is shown in fig.20 . It can be seen on fig. 21 that the wheel odometry error is smaller than the quantization error. Then, in fig 22, the beginning of a catastrophic event that greatly increases wheel odometry error is shown. Such error is can be seen in fig. 23; however, dual HMM SLAM can re-estimate its pose after traversing a reliable transition, since the system resets its odometry to a known transition value, remaining operational on the long run as shown on figs. 24 and 25.

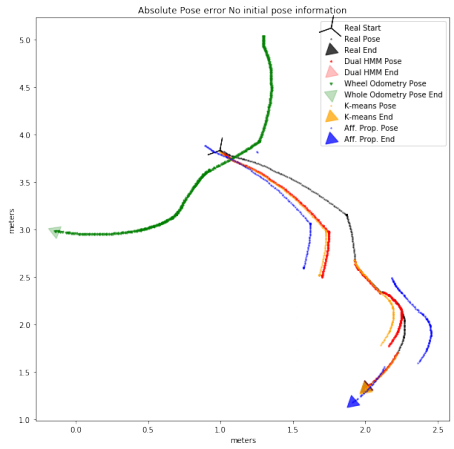


Fig. 18. In case wheel odometry has no initial information.

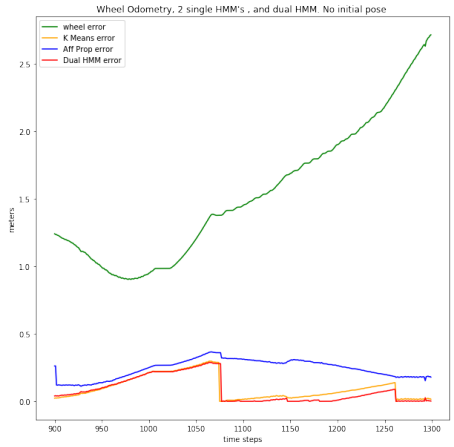


Fig. 19. Absolute pose error of run with unknown initial conditions.

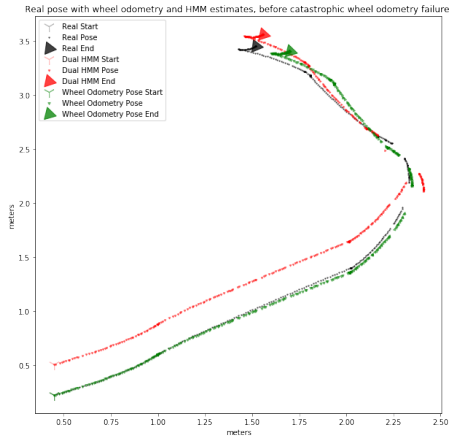


Fig. 20. Wheel Odometry and Dual-HMM error. Event start.In black the real pose sequence, green is wheel odometry with initial conditions, in red the Dual-HMM, which has a quantization error at the beginning of the run. Before a catastrophic event.

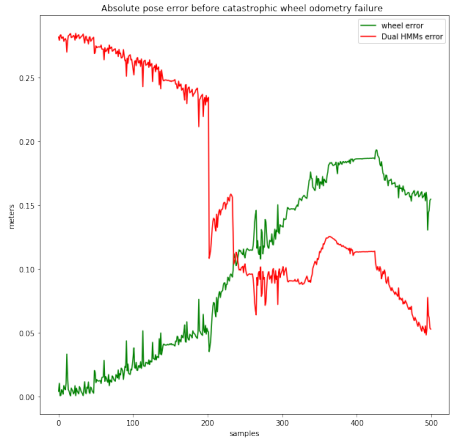


Fig. 21. Absolute pose error of wheel odometry and Dual-HMM. Both errors are relatively small, it is interesting to see how after a few time steps Dual HMM error is similar.

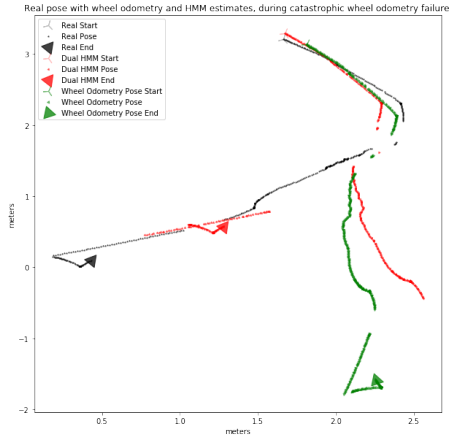


Fig. 22. Wheel Odometry in green and Dual-HMM in red during Event

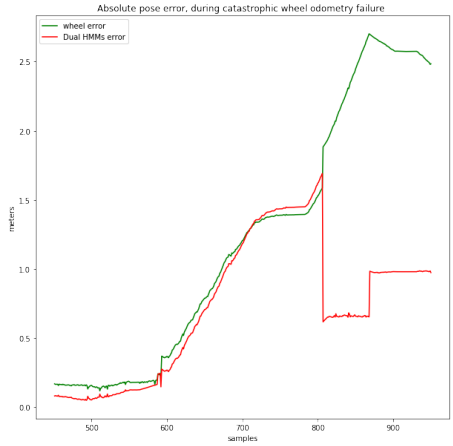


Fig. 23. Wheel Odometry and Dual-HMM error. During Event

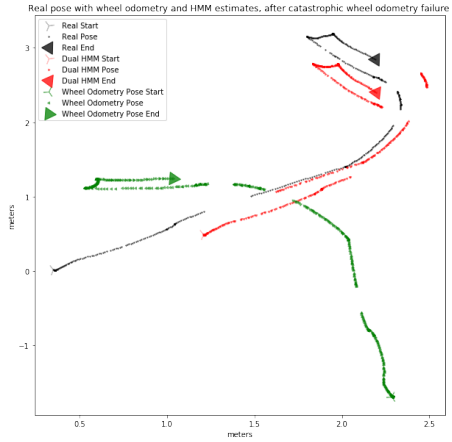


Fig. 24. Wheel Odometry and Dual-HMM estimates after Event

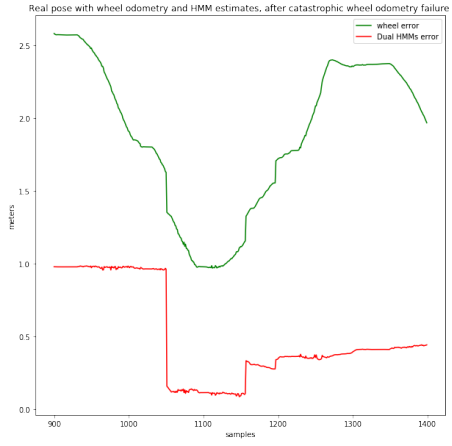


Fig. 25. Wheel Odometry and Dual-HMM error. After Event

6. Conclusions and Future Work.

We proposed a SLAM method based on Hidden Markov Models which generated a modular graphe representation of the environment; once the model was calculated, the kidnapped robot problem was solved in a number of environments using 2D sensor readings as observations. Also, although it was not shown here, the localization system remained operational during different home service tasks that required robot navigation in crowded environments. The experiments suggest that SLAM was performed successfully; it is important to notice that the online implementation of the algorithms made it possible for real-time optimization of the graphe representation.

The same measurements were quantized combining different symbols from the same source using two clustering methods. The same principle could also be applied to a wide variety of sensors or different features extracted from the same sensor measurements. This makes our system not only robust but versatile, as it can be easily adapted to use a wide variety of sensors and action policies.

Future research is being conducted with aims in an active SLAM implementation using sensor fusion of laser, 2D image features, CNN extracted features, and 3D map features (octomaps [31]). At the same time, the optimal action policy to navigate the environment will be found using Q-learning or other appropriate reinforcement learning techniques.

References

1. Takashi Yamamoto, Tamaki Nishino, Hideki Kajima, Mitsunori Ohta, and Koichi Ikeda. Human {Support} {Robot} ({HSR}). In *{ACM} {SIGGRAPH} 2018 {Emerging} {Technologies}*, {SIGGRAPH} '18, pages 11:1—-11:2, New York, NY, USA, 2018. ACM.
2. P Zarchan and H Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. Number v. 190 in Fundamentals of Kalman filtering: a practical approach. American Institute of Aeronautics and Astronautics, Incorporated.
3. R Smith, M Self, and P Cheeseman. Autonomous Robot Vehicles. chapter Estimating, pages 167–193. Springer-Verlag, Berlin, Heidelberg, 1990.
4. Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.
5. D Fox, W Burgard, F Dellaert, S Thrun AAAI/IAAI, and undefined 1999.
6. G Grisetti, R Kuemmerle, C Stachniss, and W Burgard. A Tutorial on Graph-Based {SLAM}. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.
7. M. Zaffar, S. Ehsan, R. Stolkin, and K. M. Maier. Sensors, slam and long-term autonomy: A review. In *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 285–290, 2018.
   Monte carlo localization: Efficient position estimation for mobile robots. *aaai.org*, 1999.
8. Sebastian Thrun and Michael Montemerlo. The graphe SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.
9. A. Soragna, M. Baldini, D. Joho, R. Kümmerle, and G. Grisetti. Active slam using connectivity graphs as priors. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 340–346, 2019.
10. João Machado Santos, David Portugal, and Rui P. Rocha. An evaluation of 2d slam techniques available in robot operating system. In *SSRR*, pages 1–6. IEEE, 2013.
11. Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.
12. Regis Vincent, Benson Limketkai, and Michael Eriksen. Comparison of indoor robot localization techniques in the absence of GPS. In Russell S Harmon, John H Holloway Jr., and J Thomas Broach, editors, *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV*, volume 7664, pages 606–610. International Society for Optics and Photonics, SPIE, 2010.
13. Kurt Konolige, Giorgio Grisetti, Rainer Kümmerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent. Efficient sparse pose adjustment for 2d mapping. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 22–29, 2010.
14. G O S Ekhaguere. On notions of Markov property. *Journal of Mathematical Physics*, 18(11):2104–2107, 1977.
15. Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989.
16. D Fox, W Burgard, F Dellaert, S Thrun AAAI/IAAI, and undefined 1999. Monte carlo localization: Efficient position estimation for mobile robots. *aaai.org*, 1999.
17. Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4), 1986.
18. Savage , Fuentes Map representation using hidden markov models for mobile robot localization. volume 161, 2018.

19. A P Dempster, N M Laird, and D B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.
20. Pierre Baldi and Yves Chauvin. Smooth {On}-{Line} {Learning} {Algorithms} for {Hidden} {Markov} {Models}. *Neural Comput.*, 6, 1993.
21. Long Wen, Yang Zhao, Shuguang Li, Hong Cheng, and Chen Zhang. {MST}-{ResNet}: {A} {Multiscale} {Spatial} {Temporal} {ResNet} for {Steering} {Prediction}. pages 246–251, 2019.
22. Jianbo Shi and Carlo Tomasi. Good Features to Track. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.
23. Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315:972–977, 2007.
24. Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
25. G D Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
26. Agostino Martinelli. Modeling and {Estimating} the {Odometry} {Error} of a {Mobile} {Robot}. *IFAC Proceedings Volumes*, 34(6):407–412, 2001.
27. Yasuyoshi Yokokohji, Yoshihiro Kawai, Mizuho Shibata, Yasumichi Aiyama, Shinya Kotosaka, Wataru Uemura, Akio Noda, Hiroki Dobashi, Takeshi Sakaguchi, and Kazuhito Yokoi. World robot summit – summary of the pre-competition in 2018. *Advanced Robotics*, pages 1–24, 09 2019.
28. ROS. ROS (Robot Operating System, 2018.
29. Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.
30. Imam Bukhori and Zool Ismail. Detection of kidnapped robot problem in monte carlo localization based on the natural displacement of the robot. *International Journal of Advanced Robotic Systems*, 14:172988141771746, 07 2017.
31. Kai Wurm, A Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. volume 2, 01 2010.
32. Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, and Alexander Kleiner. On Measuring the Accuracy of SLAM Algorithms. Technical report.

Fuentes Casarrubias Oscar — M.Eng. in electrical engineering, lecturer; team leader of HSRB team in Bio Robotics Laboratory. Research interests: Machine Learning, Deep learning, Computer Vision , Artificial Intelligence, SLAM, navigation.The number of publications — 3. ofc80@comunidad.unam.mx Bio Robotics Laboratory, School of Engineering.Circuito Exterior S/N, Ciudad Universitaria, 04510, Mexico City, Mexico; office phone: +52(55)56223041.
Savage Carmona Jesus — Ph. D.;Professor, Electrical Engineering Department, National Autonomous University of Mexico (UNAM). Research interests: autonomous mobile robots, digital signal processing, computer architectures. robotssavage@gmail.com; Circuito Exterior S/N, Ciudad Universitaria, 04510, Mexico City, Mexico; office phone: +52(55)56223041.
Contreras Luis — Ph. D.; Received his Ph.D. in Computer Science at the Visual Information Laboratory, in the Department of Computer Vision, University of Bristol, UK, working on problems related to computer vision and robotics; currently, Adjunct Research Fellow at the Advanced Intelligence and Robotics Research Center, Tamagawa University, Japan. contreras.luis@lab.tamagawa.ac.jp; 6-1-1 Tamagawagakuen, Machida, Tokyo, 194-8610, Japan.

Acknowledgements. This research is supported by Concayt.

ФуэнтесОСКАР , Сэвидж ХЕСУС , Контрерас ЛУИС

СИСТЕМА SLAM, ОСНОВАННАЯ НА СКРЫТЫХ МАРКОВСКИХ МОДЕЛЯХ

Фуэнтес Оскар Сэвидж Хесус Контрерас Луис **Система SLAM, основанная на скрытых марковских моделях.**

Аннотация. Мы представляем графическую систему SLAM, основанную на скрытых марковских моделях (HMM), где показания датчиков представлены различными символами с использованием ряда методов кластеризации; затем символы объединяются как единое предсказание для повышения точности с использованием Dual HMM. Универсальность нашей системы позволяет ей работать с различными типами датчиков и реализовывать, активный или пассивный, графическое SLAM. Робот Toyota HSR (Human Support Robot) использовался для создания набора данных как в реальных, так и в смоделированных условиях соревнований. Мы протестировали нашу систему в задаче о похищении роботов, обучив представление, улучшив его в режиме онлайн и, наконец, решив проблему SLAM. В этой статье представлена система SLAM с активным графом, основанная на скрытых марковских моделях (HMM). Для создания набора данных использовались измерения датчика Hokuyo робота Toyota Human Support Robot HSR. Эта система использует показания лазера для генерации набора данных, используемого HMM. Двойная реализация HMM использует преимущества универсальности метода для реализации объединения функций, используя подход «лучшее из двух миров». Метод был протестирован как на реальных, так и на смоделированных аренах World Robot Summit и Robcup.

Ключевые слова: SLAM, навигация, локализация, картографирование, машинное обучение, Markov, Lidar, Resnet, Sensor Fusion, автономный робот, сервисный робот, Toyota HSR, grpahe, скрытая марковская модель, gmapping, картограф

Оскар Фуэнтес Касаррубиас — M.Eng. по электротехнике, доцент; руководитель группы HSRB в лаборатории биоробототехники. Область научных интересов: машинное обучение, глубокое обучение, компьютерное зрение, искусственный интеллект, SLAM, навигация. Количество публикаций — 3. ofc80@comunidad.unam.mx Лаборатория биоробототехники, UNAM, Школа инженерии, здание T. Мехико

Саваж Хесус — Доктор философии; профессор кафедры электротехники Национального автономного университета Мексики (НАУМ). Область научных интересов: автономные мобильные роботы, цифровая обработка сигналов, компьютерные архитектуры. robotssavage@gmail.com; Циркуито Экстериор б/н, Университетский город, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Литература

1. Такаши Ямамото, Тамаки Нишино, Хидеки Кадзима, Мицунори Охта и Коичи Икеда.
2. П Зарчан и Х Мусофф. *Основы фильтрации Калмана: практический подход*. Number v. 190 в Основах калмановской фильтрации: практический подход. Американский институт аэронавтики и астронавтики, Incorporated.
3. Р Смит, М Селф и П Чизмен. Автономные автомобили-роботы. глава Оценка, страницы 167–193. Шпрингер-Верлаг, Берлин, Гейдельберг, 1990.
4. Джорджио Гризетти, Сирилл Стахнисс и Вольфрам Бургард. Улучшенные методы построения сеток с помощью частиц Рао-Блэквелла Фильтры. *IEEE Transactions on Robotics*, 23 (1): 34–46, 2007.

5. D Fox, W Burgard, F Dellaert, S Thrun AAAI / IAAI и undefined 1999.
6. G Grisetti, R Kuemmerle, C Stachniss и W. Burgard. Учебник по графическому SLAM . *Intelligent Transportation Systems Magazine, IEEE*, 2 (4): 31–43, 2010 г.
7. М. Заффар, С. Эхсан, Р. Столкин и К. М. Майер. Датчики, шум и длительная автономность: обзор. In *Конференция NASA / ESA 2018 по адаптивному оборудованию и системам (AHS)*, страницы 285-290, 2018.
Локализация Монте-Карло: эффективное определение местоположения для мобильных устройств роботы. *aaai.org*, 1999.
8. Себастьян Трун и Майкл Монтемерло. Алгоритм graphe SLAM с приложениями к крупномасштабному отображению Городские сооружения. *Международный журнал исследований робототехники*, 25 (5-6): 403-429, 2006.
9. А. Сорагна, М. Балдини, Д. Йохо, Р. Кюммерле и Г. Гризетти. Активный слэм с использованием графов связности в качестве априорных значений. In *Международная конференция IEEE / RSJ по интеллектуальным роботам, 2019 г. and Systems (IROS)*, страницы 340–346, 2019.
10. Жуан Мачадо Сантос, Давид Португалия и Руи П. Роча. Оценка методов двумерного удара, доступных в работе робота. система. В *SSRR*, страницы 1–6. IEEE, 2013.
11. Джорджио Гризетти, Сирилл Стахнисс и Вольфрам Бургард. Улучшенные методы построения сеток с помощью частиц Рао-Блэквелла Фильтры. *IEEE Transactions on Robotics*, 23 (1): 34–46, 2007.
12. Регис Винсент, Бенсон Лимкеткай и Майкл Эриксен. Сравнение методов локализации комнатных роботов при отсутствии GPS. In Russell S Harmon, John H Holloway Jr. и J Thomas Broach, редакторы, *Обнаружение и обнаружение мин, взрывоопасных предметов и скрытых объектов Targets XV*, том 7664, страницы 606-610. Международное общество оптики и фотоника, SPIE, 2010.
13. Курт Конолиге, Джорджио Гризетти, Райнер Кюммерле, Вольфрам Бургард, Бенсон Лимкеткай и Регис Винсент. Эффективная корректировка разреженной позы для 2d маппинга. На *Международной конференции IEEE / RSJ 2010 по интеллектуальным роботам and Systems*, страницы 22–29, 2010 г.
14. G O S Ekhaguere. О понятиях марковского свойства. *Journal of Mathematical Physics*, 18 (11): 2104–2107, 1977.
15. Лоуренс Р. Рабинер. Учебное пособие по скрытым марковским моделям и избранным приложениям в распознавание речи. В *PROCEEDIES OF THE IEEE*, страницы 257–286, 1989.
16. D Fox, W Burgard, F Dellaert, S Thrun AAAI / IAAI и undefined 1999. Локализация Монте-Карло: эффективное определение местоположения для мобильных устройств роботы. *aaai.org*, 1999.
17. Рэндалл С. Смит и Питер Чизмен. О представлении и оценке пространственной неопределенности. *Международный журнал исследований робототехники*, 5 (4), 1986.
18. Сэвидж, Фуэнтес Представление карты с использованием скрытых марковских моделей для локализации мобильного робота. , том 161, 2018.
19. А. П. Демпстер, Н. М. Лэрд и Д. Б. Рубин. Максимальная вероятность получения неполных данных с помощью алгоритма EM. *ЖУРНАЛ КОРОЛЕВСКОГО СТАТИСТИЧЕСКОГО ОБЩЕСТВА, СЕРИЯ В*, 39 (1): 1–38, 1977.
20. Пьер Бальди и Ив Шовен. Smooth On-Line Learning Algorithms для Скрытых Марковских моделей .
Neural Comput., 6, 1993.

21. Лун Вэнь, Ян Чжао, Шугуан Ли, Хун Чэн и Чэнь Чжан. MST-ResNet : A Мультимасштаб Spatial Temporal ResNet для рулевого управления
22. Джианбо Ши и Карло Томази. Хорошие возможности для отслеживания. em Конференция IEEE по компьютерному зрению и распознаванию образов, страницы 593-600, 1994.
23. Брендан Дж. Фрей и Делберт Дук. Кластеризация путем передачи сообщений между точками данных. em Science, 315: 972–977, 2007.
24. Мин Лян, Бинь Ян, Юн Чен, Жуй Ху и Ракель Уртасун. Многозадачная мультисенсорная комбинация для обнаружения трехмерных объектов. В em Proceedings of the IEEE / CVF Conference on Computer Vision and Распознавание образов (CVPR), июнь 2019 г.
25. Б-г Форни. Алгоритм Витерби. em Proceedings of the IEEE, 61 (3): 268–278, 1973.
26. Агостино Мартинелли. Моделирование и оценка ошибки одометрии мобильного робота }. em IFAC Proceedings Volumes, 34 (6): 407–412, 2001.
27. Ясуёси Ёкокоджи, Ёсихиро Каваи, Мидзухо Сибата, Ясумичи Айяма, Шинья Котосака, Ватару Уэмура, Акио Нода, Хироки Добаши, Такеши Сакагути и Кадзухито Ёкои. World robot Summit - итоги предсоревнований 2018. em Advanced Robotics, страницы 1–24, 09, 2019.
28. РОС. ROS (операционная система роботов, 2018.
29. Натан Кениг и Эндрю Ховард. Разработка и использование парадигм для беседки, мульти-робота с открытым исходным кодом симулятор. In em Международная конференция IEEE / RSJ по интеллектуальным роботам и Systems, страницы 2149-2154, Сендай, Япония, сентябрь 2004 г.
30. Имам Бухори и Зоол Исмаил. Проблема обнаружения похищенного робота в локализации монте-карло на основе естественного перемещения робота. em Международный журнал передовых робототехнических систем, 14: 172988141771746, 07 2017.
31. Кай Вурм, А Хорнунг, Марен Бенневиц, Сирилл Стахнисс и Вольфрам Бургард. Octomap: вероятностное, гибкое и компактное представление трехмерной карты для робототехнических систем. volume 2, 01 2010.
32. Бастиан Стедер, Кристиан Дорнхеге, Майкл Рунке, Джорджио Гризетти, Сирилл Стахнисс и Александр Клейнер. Об измерении точности алгоритмов SLAM. Технический отчет.

Bibliografía

[1] "Map of california as an island (1650)."

[2] T. Yamamoto, T. Nishino, H. Kajima, M. Ohta, and K. Ikeda, "Human {Support} {Robot} ({HSR})," in *ACM SIGGRAPH 2018 EmergingTechnologies*, {SIGGRAPH} '18, (New York, NY, USA), pp. 11:1—-11:2, ACM, 2018.

[3] "Map representation using hidden markov models for mobile robot localization," vol. 161, 2018.

[4] I. Bukhori and Z. Ismail, "Detection of kidnapped robot problem in monte carlo localization based on the natural displacement of the robot," *International Journal of Advanced Robotic Systems*, vol. 14, p. 172988141771746, 07 2017.

[5] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach*. No. v. 190 in Fundamentals of Kalman filtering: a practical approach, American Institute of Aeronautics and Astronautics, Incorporated.

[6] R. Smith, M. Self, and P. Cheeseman, "Autonomous Robot Vehicles," ch. Estimating, pp. 167–193, Berlin, Heidelberg: Springer-Verlag, 1990.

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[8] G. Grisetti, R. Kuemmerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based {SLAM}," *Intelligent Transportation Systems Magazine, IEEE*, vol. 2, no. 4, pp. 31–43, 2010.

[9] M. Zaffar, S. Ehsan, R. Stolkin, and K. M. Maier, "Sensors, slam and long-term autonomy: A review," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 285–290, 2018.

[10] S. Thrun and M. Montemerlo, "The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[11] A. Soragna, M. Baldini, D. Joho, R. Kümmerle, and G. Grisetti, "Active slam using connectivity graphs as priors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 340–346, 2019.

[12] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2d slam techniques available in robot operating system.," in *SSRR*, pp. 1–6, IEEE, 2013.

[13] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[14] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of indoor robot localization techniques in the absence of GPS," in *Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV* (R. S. Harmon, J. H. H. Jr., and J. T. Broach, eds.), vol. 7664, pp. 606–610, International Society for Optics and Photonics, SPIE, 2010.

[15] G. O. S. Ekhaguere, "On notions of Markov property," *Journal of Mathematical Physics*, vol. 18, no. 11, pp. 2104–2107, 1977.

[16] P. Baldi and Y. Chauvin, "Smooth on-line learning algorithms for hidden markov models," *Neural Comput.*, vol. 6, 1993.

[17] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[19] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *PROCEEDINGS OF THE IEEE*, pp. 257–286, 1989.

[20] F. O., "Localización de un robót móvil utilizando Modelos Ocultos de Markov," Master's thesis, Posgrado de Ingeniería UNAM, Junio 2018.

[21] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84–95, 1980.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: {Machine} {Learning} in {Python}," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] G. Capraro, G. Berdan, R. Liuzzi, and M. Wicks, "Artificial intelligence and sensor fusion," in *IEMC '03 Proceedings. Managing Technologically Driven Organizations: The Human Side of Innovation and Change (IEEE Cat. No.03CH37502)*, pp. 591–595, 2003.

[24] D. Canin, *F-35 High Angle of Attack Flight Control Development and Flight Test Results.*

[25] J. Z. Sasiadek, "Sensor fusion," *Annual Reviews in Control*, vol. 26, no. 2, pp. 203–228, 2002.

[26] M. A. Arbib, "The metaphorical brain. an introduction to cybernetic as artificial intelligence and brain theory," *Revista Portuguesa de Filosofia*, vol. 31, no. 3, pp. 331–331, 1975.

[27] L. Wen, Y. Zhao, S. Li, H. Cheng, and C. Zhang, "{MST}-{ResNet}: {A} {Multiscale} {Spatial} {Temporal} {ResNet} for {Steering} {Prediction}," pp. 246–251, 2019.

[28] B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On Measuring the Accuracy of SLAM Algorithms," tech. rep.

[29] J. Hernández, J. Savage, M. Negrete, L. Contreras, C. Sarmiento, O. Fuentes, and H. Okada, "Sparse-Map: automatic topological map creation via unsupervised learning techniques," *Advanced Robotics*, vol. 36, no. 17-18, pp. 825–835, 2022.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 39, no. 1, pp. 1–38, 1977.

[31] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[32] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, 1986.

[33] Y. Yokokohji, Y. Kawai, M. Shibata, Y. Aiyama, S. Kotosaka, W. Uemura, A. Noda, H. Dobashi, T. Sakaguchi, and K. Yokoi, "World robot summit – summary of the pre-competition in 2018," *Advanced Robotics*, pp. 1–24, 09 2019.

[34] R. Federation, "RoboCup@Home," 2016.

[35] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," vol. 2, 01 2010.