Exploración integrada probabilista para robots móviles en ambientes complejos

Alfredo Toriz Palacios[1] y Abraham Sánchez López[2]

[1] Le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, Francia

[2] Departamento de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla, México

alfredo.torizpalacios@lirmm.fr, asanchez@cs.buap.mx

Resumen. La utilización de mapas de ambiente, es uno de los requisitos principales para casi todas las tareas en robótica móvil. Desafortunadamente, no siempre es posible contar con este elemento, ya sea por la inaccesibilidad de los entornos o simplemente porque no se cuenta con una descripción utilizable de él. La solución a este problema es conocida como Exploración Integrada o SPLAM (Planificación, localización y mapeo simultáneos). Considerando este problema, presentamos algunas estrategias basadas en el filtro de Kalman extendido donde un robot móvil construye incrementalmente un mapa de su ambiente mientras simultáneamente usa este mapa para estimar la localización absoluta del robot. Al mismo tiempo, las decisiones locales sobre a donde debe moverse son realizadas utilizando la estrategia probabilística SRT, con la finalidad de minimizar el error en la estimación de la pose móvil. Aunque las estrategias basadas en herramientas clásicas mostraron buenos resultados, algunos problemas inherentes a las metodologías impiden obtener resultados óptimos. Por esta razón, nuestro trabajo se ha enfocado en la creación de una estrategia de SPLAM en donde la localización y el mapeo simultáneo se realizan usando un enfoque topológico radical basado en curvas *B-Splines*, las cuales han permitido la representación de ambientes complejos no estructurados. Así mismo, la planificación de la exploración es dirigida utilizando una nueva estrategia probabilística basada en grafos. Finalmente, los resultados obtenidos con la nueva solución propuesta son validados comparándolos con los resultados obtenidos mediante las estrategias basadas en herramientas clásicas.

Palabras clave. SPLAM, exploración integrada, SLAM, curvas *B-Splines.*

Probabilistic Integrated Exploration for Mobile Robots in Complex Environments

Abstract. The use of environment maps is one of the main requirements for almost all tasks in mobile robotics. Unfortunately, it is not always possible to have this element, either by the inaccessibility of the environment or simply because there is no usable description of it. The solution to this problem is known as Integrated Exploration or SPLAM (Simultaneous Planning, Localization and Mapping). Considering this problem, we present some strategies based on the extended Kalman filter (EKF), when the mobile robot incrementally builds a map of its environment, while simultaneously using this map for computing the absolute robot localization. At the same time, local decisions on where to move next are performed using the probabilistic strategy–SRT, in order to minimize the error of the estimation of the robot's pose and the configuration locations. Although the classic strategies have shown good results, there are some inherent problems that prevent achieving the optimal results. For this reason, the paper has focused on creating a SPLAM strategy when the simultaneous localization and mapping is performed using a radical topological approach based on B-spline curves and when the planning of the exploration is conducted using a probabilistic strategy based on graphs.

Keywords. SPLAM, integrated exploration, SLAM, B-Spline curves.

1. Introducción

Uno de los retos fundamentales de la robótica actual es el de obtener mecanismos robustos y eficientes para modelar ambientes cada vez más complejos empleando robots móviles para su exploración. El enfoque clásico manejado para este fin es conocido como el problema de Localización y Mapeo Simultaneo (SLAM), el cual consiste en utilizar el mapa que el robot está actualmente construyendo para determinar su propia posición. Este problema puede ser técnicamente desafiante porque la posición del robot y las características del ambiente deben ser estimadas simultáneamente a partir de los datos con ruido obtenidos del sensor láser de rango embarcado en el robot.

Las soluciones probabilísticas son las más populares para el problema de SLAM, especialmente aquellas basadas en el filtro de Kalman extendido para estimar el mapa. Sin embargo, a pesar de toda la investigación hecha en este campo, la cual ha resultado en un progreso substancial en la creación de mapas, aún existen barreras significativas que surgen en la implementación de estos algoritmos. El primero viene dado, porque normalmente los algoritmos de SLAM asumen implícitamente un control ingenuo, donde la intervención humana conduce al robot a través del ambiente mientras este registra los datos del sensor. Esto da como resultado un sistema que no es realmente autónomo e incluso la calidad del mapa es pobre cuando los datos del sensor son obtenidos a partir de un robot que es controlado manualmente por una persona inexperta.

Dado lo anterior, es necesario considerar el concepto de "exploración de ambientes desconocidos", el cual responde a la pregunta de a dónde debe moverse el robot a continuación, con la finalidad de construir un mapa óptimo del ambiente.

Aunque la estrategia de exploración puede tener un impacto real en la calidad del mapa resultante, el área de exploración para SLAM es relativamente nueva y poco usada. Las estrategias en esta área, requieren algoritmos rápidos que puedan adaptarse a cambios en el ambiente cuando se detectan nuevas características u obstáculos.

La integración de estrategias de exploración al paradigma clásico de SLAM, ha dado paso a un nuevo enfoque dirigido al problema de construcción de mapas de forma autónoma denominado *Exploración Integrada* o *SPLAM (Simultaneous Planning, Localization and Mapping).*

Un segundo problema, es con respecto a la representación de objetos en el mapa y la información que tal representación pueda proporcionar al proceso de SLAM. Las curvas *B-Splines* han sido recientemente usadas [11] como una forma de representación de ambientes. En este trabajo, estas curvas han mostrado gran eficiencia y versatilidad para describir ambientes complejos donde no es posible extraer características simples, tales como líneas y puntos. Sin embargo, el uso de este tipo de representación es demasiado reciente y por lo tanto no completamente explotado, especialmente tomando en cuenta el tipo de información que las *B-Splines* pueden proporcionar al proceso de SLAM.

Finalmente, las limitaciones en el tamaño del ambiente debido a los cálculos que deben ser realizados y las inconsistencias que las linearizaciones del problema producen en algunos métodos, hacen necesario encontrar formas alternativas de abordar el problema de SLAM.

Dado lo anterior, el objetivo principal de este trabajo, es desarrollar herramientas efectivas y robustas en el campo de la planificación, localización y mapeo, que coexistan armoniosamente para obtener una solución destinada a la construcción de ambientes complejos de forma autónoma. Así mismo, hemos creado enfoques de SPLAM basados en herramientas clásicas para comprobar de primera mano las limitaciones de estos algoritmos y al mismo tiempo utilizarlas para validar nuestra propuesta.

Considerando las motivaciones y el objetivo planteado, el documento está organizado como sigue. La sección 2 introduce una aproximación al problema de exploración integrada usando algunas herramientas bien conocidas en el campo de la exploración de ambientes y del propio SLAM. El objetivo final de esta sección, es el de construir una estrategia de SPLAM, la cual

Computación y Sistemas Vol. 18 No. 1, 2014 pp. 195-215
ISSN 1405-5546
http://dx.doi.org/10.13053/CyS-18-1-2014-028

servirá como comparación para validar la aproximación desarrollada en este proyecto y que se presentará en la sección 3.

La sección 3 representa la sección principal de esta propuesta, en ella presentamos el desarrollo de nuevos métodos tanto en el área de exploración de ambientes como en el área de SLAM, para obtener una nueva estrategia de Exploración Integrada. Así, la exploración es dirigida por un método de exploración basado en un grafo de exploración y la construcción del mapa se efectúa mediante un método de SLAM topológico basado en curvas *B-Splines*.

La sección 4 contiene los resultados que muestran y apoyan la aplicación práctica de los métodos, así como la comparación necesaria para mostrar la validez de nuestra proposición comparándola con las metodologías presentadas en la sección 2. Los resultados incluyen simulaciones llevadas a cabo en entornos construidos, así como experimentos en entornos reales.

Finalmente, la sección 5 presenta las principales contribuciones del trabajo y las conclusiones que se pueden extraer de ellas. Además, se realizará un análisis de la propuesta, teniendo en cuenta las posibles mejoras para un trabajo futuro y las posibles extensiones del proyecto.

2. Enfoque SRT-EKF

2.1. Exploración SRT

Una de las principales tareas del problema de SPLAM es navegar a través del ambiente con la finalidad de construir un mapa de una manera realmente autónoma. Siguiendo este pensamiento, en esta sección hemos usado una poderosa herramienta conocida como el "Árbol aleatorio basado en sensor" (SRT) presentado por Oriolo *et al.* [10].

El método SRT está basado en la generación aleatoria de configuraciones del robot dentro de un área de seguridad local detectada por los sensores. Estas configuraciones son almacenadas en una estructura de datos tipo árbol que representa el roadmap del área explorada y la región de seguridad asociada (SR). Cada nodo del árbol (T), consta de una posición de robot y su región de seguridad local (LSR) que es construida a través del sistema de percepción del robot. Esta LSR, es un estimado del espacio libre circundante al robot en una configuración dada.

La Figura 1, muestra cómo trabaja el método SRT. Al inicio de cada iteración, el algoritmo obtiene la LSR asociada con la posición actual del robot, q_{curr}. Una vez obtenida la LSR, la función **EXTEND_TREE** es responsable de actualizar el árbol agregando la posición del robot y su correspondiente LSR a cada nodo. Después, el ambiente global almacenado (Amb_B*S*) será actualizado con las nuevas características extraídas de la LSR que aún no forman parte de él. Este proceso es realizado por el procedimiento **UPDATE.**

El siguiente paso consiste en procesar el límite local *F*, es decir, identificar obstáculos y áreas libres. Generalmente, *F* es una colección de arcos discretos. Después de obtener estos límites y si aún existen zonas libres, el procedimiento RANDOM_DIR generará direcciones aleatorias con el objetivo de elegir

```
INTEGRATED_EXPLORATION (q_init, k_max)
1.  q_curr ← q_init;
2.  for k=1 to K_max
3.      S ← LSR(q_curr);
4.      T ← EXTEND_TREE(q_curr,S,T);
5.      Amb_BS ← UPDATE (S);
6.      F ← FRONTIER(q_curr,S) ;
7.      if F≠0
8.          i ← 0;
9.          VALID ← FALSE;
10.         While ((i<MAX_ITER) && (!VALID))
11.                 θ_rand ← RANDOM_DIR(F);
12.                 q_cand ← DISPLACE(q_cur, θ_rand);
13.                 VALID ← VALID_CONF(q_cand);
14.                 i++;
15.         end
16.         if (VALID )
17.                 q_dest ← q_cand;
18.         else
19.                 q_dest ← q_curr.parent;
20.                 if q_dest=NULL
21.                     return [T,Amb_BS];
22.                 end
23.         end
24.     else
25.         q_dest ← q_curr.parent;
26.         if q_dest=NULL
27.                 return [T,Amb_BS];
28.         end
29.     end
30.     MOVE_TO(q_dest, q_curr, Amb_BS );
31.     q_curr ← q_dest;
32. end
33. return [T,Amb_BS];
```

Fig. 1. Algoritmo de exploración integrada basada en SRT

una que apunte hacia un arco libre, entonces, se generará una configuración q_{cand} dando un paso de longitud α en la dirección θ_{rand} elegida. El tamaño del paso α es elegido como una fracción fija del radio de la LSR en esa dirección particular. Debido a la forma de S, q_{cand} estará libre de colisión. En caso de que ningún arco esté libre, el robot regresará a la posición del nodo padre de q_{curr} y el ciclo de exploración comenzará de nuevo.

Una vez que la configuración q_{cand} es obtenida, el procedimiento **VALID_CONF** asegurará que esta nueva configuración sea válida, es decir, que esta nueva configuración se encuentre fuera de las LSRde los otros nodos en el árbol. Si esta nueva configuración es válida, será el nuevo destino q_{dest} que el robot debe alcanzar. En el caso contrario, si después de un número máximo de intentos no es posible encontrar una configuración q_{cand}, el nodo padre será la nueva configuración q_{dest} (es decir, el robot regresará al padre de la actual configuración del nodo).

Después de que la configuración q_{dest} es obtenida, la función MOVE_TO (Figura 2) permite al robot moverse a esta nueva configuración. El proceso se realiza buscando en la lista de entradas de control (*list_U*), una entrada $u_{control}$ que permita al robot aproximarse a q_{dest} desde la posición q_{curr} (función **BEST_U**). Una vez elegida la mejor entrada $u_{control}$, esta se aplica al robot.

En este punto, se obtienen la posición odométrica y los incrementos en X, Y y θ entre la posición previa y la posición actual odométrica (ΔX, ΔY, Δθ). La información reportada por el robot será esencial para obtener la posición estimada que será usada por el método **LOCALIZATION** para obtener la posición real. Este algoritmo se repite hasta que las configuraciones q_{curr} y q_{dest} sean la misma.

2.2. EKF-SLAM clásico

La exploración de ambientes desconocidos requiere de una funcionalidad adicional debido a que la información odométrica reportada por el robot, en la mayoría de los casos no es exacta, resultando en mapas inexactos para futuras navegaciones.

```
MOVE_TO (q_curr, q_dest, , Amb_BS)

1. while q_curr != q_dest
3.       u_control ← BEST_U(List_U, q_curr, q_dest);
4.       ROBOT ← u_control;
5.       q̂ ← ODOMETRY;
6.       q_curr ← LOCALIZATION(q̂, q_curr, Amb_BS);
7. end
8. return q_curr
```

Fig. 2. Método MOVE_TO del algoritmo de exploración integrada basado en SRT

```
LOCALIZATION_EKF(q̂, X̂_{k-1}, LandM_Amb, P_{k-1})

1. [X̂_k^-, P_k^-] ← PREDICTION(q̂, X̂_{k-1}, P_{k-1});
2. D ← DATA_SEGMENTATION(X̂_k^-);
3. Ss ← LANDMARKS_EXTRACTION(D);
4. [LM_asoc, LM_new] ← LANDMARKS_ASSOCIATION(Ss, LandM_Amb);
5. [X̂_k, P_k] ← UPDATE (LM_asoc, q_est)
6. return X̂_k, P_k
```

Fig. 3. Algoritmo de localización EKF

2.2.1. Localización EKF

El algoritmo propuesto asume que la posición inicial del robot está bien localizada y, consecuentemente, la primera observación del ambiente tiene una localización perfecta.

Una vez que el robot se ha movido de una posición qlast a una posición qcurr, la nueva posición del robot se obtiene agregando a la última posición localizada los incrementos ΔX, ΔY y Δθ reportados por el sistema odométrico del robot. Después de que esta posición es estimada, el robot recolectará la información del ambiente circundante para el proceso de localización.

Finalmente, se agregan los objetos observados que no pertenecen al mapa y aquellos que se han asociado parcialmente con los objetos contenidos en el mapa se utilizan para extender los objetos almacenados. De esta forma, se formula el algoritmo EKF mostrado en la Figura 3.

La función **PREDICTION** procesa la estimación de la nueva posición del robot usando la función de movimiento $\hat{X}_k^- = f\left(\hat{X}_{k-1}, u_{k-1}, 0\right)$. Después, la función **DATA_OBTENTION**

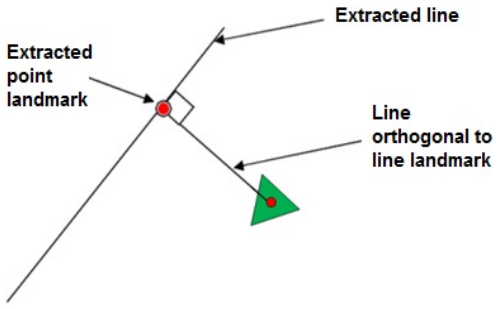


Fig. 4. Característica tipo línea como punto

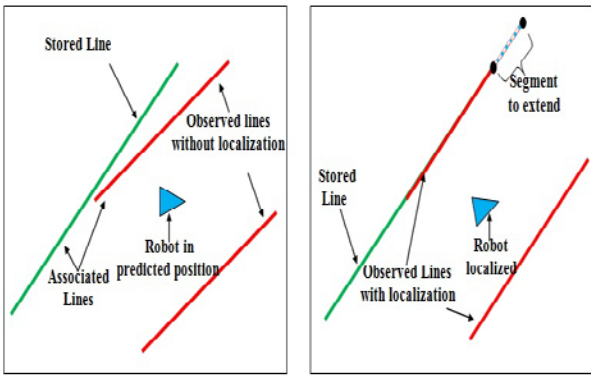


Fig. 5. Línea parcialmente asociada a ser extendida

obtendrá la información del ambiente circundante al robot que se obtiene por los sensores en el instante *k*. Esta información será colocada espacialmente en la posición actual estimada por la función **PREDICTION.**

Con la información obtenida a través de los sensores, la función **LANDMARKS_EXTRACTION** estará a cargo de buscar características del ambiente que sean fácilmente re-observables. Dado que el EKF clásico trabaja con ambientes estructurados, el sistema buscará esencialmente *líneas rectas* y *esquinas*.

Un aspecto crítico del algoritmo de localización es la asociación de datos (función **LANDMARKS_ASSOCIATION**). El objetivo de esta función es emparejar características observadas en diferentes escaneos y asignar mediciones a partir del cual se originaron. Este proceso es también conocido como re-observación de *Landmarks*.

Dados los dos tipos de *Landmarks* usados en este método, el tipo "*esquina"* es el más fácil de asociar. Para esto, se buscan grupos de características del mismo tipo almacenadas en el sistema donde la distancia euclidiana entre ellas sea menor que un umbral D_{min}.

Por otro lado, las características "*líneas rectas*" deben transformarse a puntos fijos, tomando la posición estimada del robot y calculando el punto ortogonal a las líneas rectas. Este proceso se hace tanto para las líneas observadas como para las líneas asociadas almacenadas en el sistema (Figura 4).

Si una característica presente en el vector de estado del sistema es re-observada, el paso de actualización del EKF (función UPDATE) se usa para actualizar el estado del mapa incluyendo la pose del robot.

2.2.2. Extensión del mapa

Esta sección está completamente ligada a la función UPDATE en el algoritmo de exploración integrada (Figura 1 línea 5). Una vez que se alcanzó la nueva posición a explorar q_{dest}, su nueva LSR se usará para actualizar el mapa en la variable de estado del EKF.

a) Agregar una nueva *Landmark*

Cuando se observa una nueva *landmark* y esta no se ha podido asociar, el nuevo estado de la característica se incorpora al vector estado del sistema. Para las "esquinas", sólo se agrega el ángulo y el rango hacia la característica como sigue:

$$X_{N+1} = m(\hat{X}_k, L_{new}) = \begin{bmatrix} x_{rk} + r\cos(\theta_{rk} + \theta) \\ y_{rk} + r\sin(\theta_{rk} + \theta) \end{bmatrix} \quad (1)$$

Donde m representa el mapa, *x* y *y* representan las coordenadas de la nueva característica, r es el la distancia hacia el nuevo objeto, X_n las *Landmarks* ya contenidas en el mapa y L la nueva *landmark*.

Pero para el caso de las "líneas rectas", la nueva característica tiene el ángulo y distancia del punto inicial y final, así, en este caso la nueva característica tiene la siguiente forma:

$$X_{N+1} = m(\hat{X}_k, L_{new}) = \begin{bmatrix} x_{rk} + r_s \cos(\theta_{rk} + \theta_s) \\ y_{rk} + r_s \sin(\theta_{rk} + \theta_S) \\ x_{rk} + r_f \cos(\theta_{rk} + \theta_f) \\ y_{rk} + r_f \sin(\theta_{rk} + \theta_f) \end{bmatrix} \quad (2)$$

b) Extensión de *Landmarks*

Cuando se obtiene una nueva LSR sobre una nueva posición q_{dest} alcanzada, la asociación de algunas líneas se hace sólo en forma parcial (Figura 5). En este caso, este segmento de línea se extiende con la nueva información.

Entonces, para hacer esta extensión, en primer lugar se tiene que extraer la característica correspondiente a la k-esima línea que se extenderá del vector de estado. Con esta información, se pueden calcular fácilmente los elementos *m* y *b* de la ecuación de la recta y hacer la extensión con las coordenadas x_{est} y y_{est} más lejanas de este segmento. Finalmente, la nueva línea extendida se agrega al vector del sistema en la misma posición de la cual fue extraída.

2.3. EKF-SLAM con *B-Splines*

Tomando en cuenta la creciente complejidad de las nuevas arquitecturas y formas encontradas en los ambientes donde un robot móvil podría desenvolverse, y considerando que la representación de objetos usada en el EKF clásico es válida para la construcción de ambientes estructurados con geometrías especificas, presentamos la nueva representación para modelar ambientes complejos usando curvas *B-Splines* propuesta por Pedraza *et al.* [11].

2.3.1. Localización EKF con *B-Splines*

Aunque el algoritmo empleado para la localización del robot usando el método EKF basado en *B-Splines* es el mismo que el mostrado en la Figura 3, la representación de objetos mediante curvas Splines hace indispensable utilizar técnicas y metodologías que adapten este tipo de representación al contexto del EKF. Así, en esta sección presentamos las metodologías necesarias para extraer y asociar las curvas *B-Splines* de las observaciones del robot, así como el modelo de observación que permitirá utilizar esta información en el algoritmo descrito en la sección 2.2.1.

a) Segmentación de datos y extracción de *Landmarks*

Antes de que los datos obtenidos puedan utilizarse por el algoritmo de localización, estos necesitan seguir varios procesos:

- **PRIMERA SEGMENTACIÓN**. Se debe realizar un análisis de la posición relativa de los puntos de los datos consecutivos. El objetivo es detectar puntos lo suficientemente cercanos que se supone pertenecen al mismo objeto.
- **SEGUNDA SEGMENTACIÓN.** Los segmentos obtenidos en la primera segmentación deben someterse a otra prueba para encontrar puntos consecutivos, cuyo ángulo es inferior a un umbral determinado. El objetivo de esta segmentación es detectar esquinas y curvas con curvaturas elevadas.
- **AJUSTE.** Cada uno de los obstáculos obtenidos por la segunda segmentación se ajustan a las curvas *B-Splines* de grado 3 que forman su polígono de control.

b) Asociación de *Landmarks*

Una vez que los datos del sensor se han segmentado, se realiza el proceso de asociación de datos. La primera asociación se denomina "burda" (Figura 6a), aquí los puntos de control de cada segmento obtenido en el proceso de segmentación se comparan con los puntos de control de las *Landmarks* contenidas en el mapa, usando el siguiente criterio:

$$min(dist(X_{m,i}, X_{o,j})) \leq d_{min} \quad \begin{cases} i = 1 \ldots n_m \\ j = 1 \ldots n_o \end{cases} \quad (3)$$

Así, si la distancia entre los puntos de control de la *Spline* en el mapa y los puntos de control de la *Spline* observada es menor a cierto umbral d_{min}, estos dos objetos se asociarán.

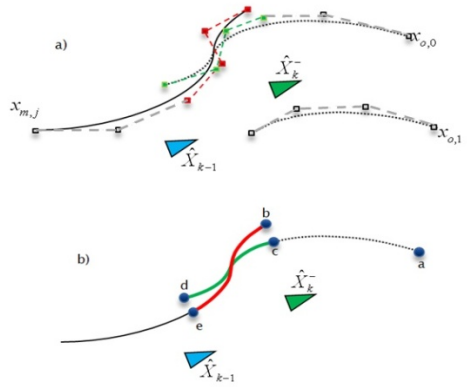


Fig. 6. Concordancia entre las curvas. a) Asociación burda. b) Asociación fina

Si ninguna *Spline* en el mapa está suficientemente cerca de la *Spline* detectada como para estar relacionada, entonces este nuevo objeto se agregará al mapa una vez que ha localizado la posición del robot. En contraste, si una *Spline* se asocia con una característica del mapa, es necesario obtener una concordancia entre los puntos (Figura 6b) como sigue:

- Uno de los extremos de la curva observada se considera como el punto *a*.
- El punto más cercano al punto a en la *Spline* contenida en el mapa se considera como el punto b.
- Si b es uno de los extremos de la *Spline* en el mapa, entonces el punto más cercano a b en la *Spline* observada se calcula y se nombra como c, si no, el punto a se asocia con el punto b.
- El proceso se repite usando el otro extremo de la *Spline* observada como punto de inicio (punto d en la Figura 6b). Este punto se asocia con el punto e de la *Spline* en el mapa.
- Gracias a la propiedad de las *B-Splines* acerca de la posibilidad de conocer la longitud de la curva, los segmentos eb y dc pueden ajustarse para tener la misma longitud.

c) Modelo de observación

Como en la sección 2.2, el uso del EKF en SLAM requiere una expresión matemática que

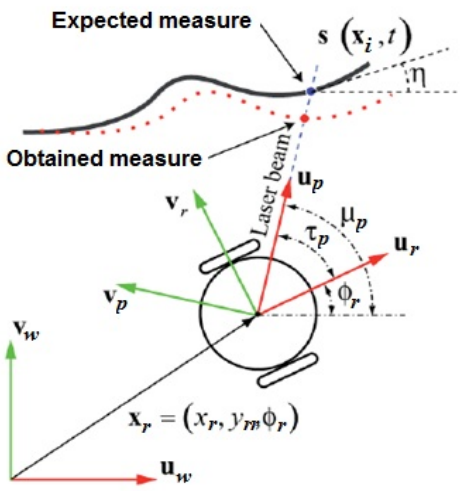


Fig. 7. Modelo de observación [12]

nos permita predecir las mediciones que esperamos obtener del sensor, dada la pose del robot y el conocimiento actual del ambiente en ese instante. El modelo de observación para el caso de las curvas *B-Splines* se reduce a encontrar la intersección de la línea recta que forma cada rayo del láser con las Splines contenidas en el mapa (Figura 7).

2.3.2. Extensión del mapa

La extensión del mapa presentada en esta sección se realiza con la función UPDATE en el algoritmo SRT (Figura 1 línea 5). Al igual que en el método clásico, la extensión del mapa se hace agregando nuevos objetos o extendiendo aquellos que ya estén contenidos en el mapa.

a) Agregar una nueva *Landmark*

Dado un sistema de mediciones $z = \{z_i,\ i = p,\ldots,\ p+q\}$ obtenidas por la posición angular del láser en el marco de referencia del robot correspondiente a la nueva característica F_{N+1} y las N características ya almacenadas en el mapa, el vector de estado se aumenta como sigue:

$$\hat{X}_k^+ = \mathrm{g}\left(\mathrm{X},\ \mathrm{Z}\right) \Leftrightarrow \begin{cases} \hat{X}_{kr} = X_r \\ \hat{X}_{ksi} = X_{si}, \quad i = 1,...,N \\ \hat{X}_{ksN+1} = \mathrm{g}_{\mathrm{sN+1}}\left(X_{\mathrm{r}},\ \mathrm{z}\right) \end{cases} \tag{4}$$

Donde X representa el mapa construido y g la función de incorporación y extensión de las nuevas características.

De esta forma, es posible obtener el vector de estado aumentado del sistema después de la inclusión de los nuevos objetos como función del estado del sistema no aumentado y de las mediciones obtenidas por el láser para este nuevo objeto.

b) Extensión de los objetos en el mapa

En el caso más frecuente, las asociaciones obtenidas serán asociadas sólo parcialmente con alguna característica del mapa (Figura 6). Esta situación indica que una nueva área inexplorada de un objeto en el mapa ha sido detectada por el sensor y, en consecuencia este nuevo conjunto de *m+1* puntos de datos no asociados deben ser integrados en el mapa.

El vector de estado resultante de la extensión del *j*-esimo objeto se obtendrá como sigue:

$$\hat{X}^+ = \mathrm{g_e}\left(X,\ \mathrm{Z}\right) \Leftrightarrow \begin{cases} \hat{X}_r^+ = X_r \\ \hat{X}_{si}^+ = X_{si} \quad ,\forall i \neq j \\ \hat{X}_{si}^+ = g_{sj}^+(X_r, X_j, Z_{q:q+m}) \end{cases} \tag{5}$$

3. Enfoque REG-Topológico

En la sección 2 hemos mostrado una aproximación para el problema de la exploración integrada (SPLAM) usando algunas herramientas bien conocidas tanto en el campo de SLAM como en el campo de la exploración de ambientes. Aunque estos algoritmos han mostrado un buen rendimiento, aún presentan algunos problemas tales como la estructura poco eficiente del método de exploración y los problemas de exactitud debido a las linearizaciones en el EKF, las cuales limitan este método a ambientes de tamaño moderado.

Lo anterior, ha motivado el desarrollo de las siguientes estrategias y procesos que mejoran el rendimiento y robustez de nuestra estrategia de SPLAM.

– El Grafo de Exploración Aleatoria (REG) es una versión modificada del algoritmo SRT. En esta propuesta, incluimos un control de fronteras para llevar a cabo un registro de los nodos que no han sido completamente explorados. Otra importante modificación es la transformación de la estructura de árbol en un grafo de exploración cuando las LSR de dos nodos sin relación se interceptan.

  La estructura del grafo junto con el control de fronteras, permitirán una exploración más eficiente dado que el robot sabe exactamente a donde ir para continuar la exploración y será capaz de desplazarse de forma óptima entre dos nodos de la estructura sin tener que pasar a través del nodo raíz cuando es necesario un cambio de rama.
– Un enfoque topológico de SLAM, se trata de un nuevo método topológico basado en curvas *B-Splines* que explota la estructura usada por el método de exploración pero también toda la información contenida en este tipo de representación.

3.1. El enfoque del grafo aleatorio de exploración

Franchi *et al*. [4], presentan en una versión modificada de su algoritmo SRT en el cual la estructura de árbol se transforma en un grafo de exploración cuando se encuentra una ruta segura para viajar entre dos nodos. Aunque nuestro método reestructura el árbol en una forma similar, la selección del siguiente nodo a explorar es completamente diferente dado que nuestro enfoque utiliza una selección aleatoria de una de las fronteras libres en el nodo actual y no una probabilidad proporcional a la longitud del árbol como en [4].

Otro aspecto importante de nuestro método es la forma en la cual los nodos con fronteras libres serán revisitados una vez que el nodo actual no tenga más fronteras libres. Esta planificación se ejecutará sólo con un conocimiento a priori de estos nodos. Por esta razón el concepto

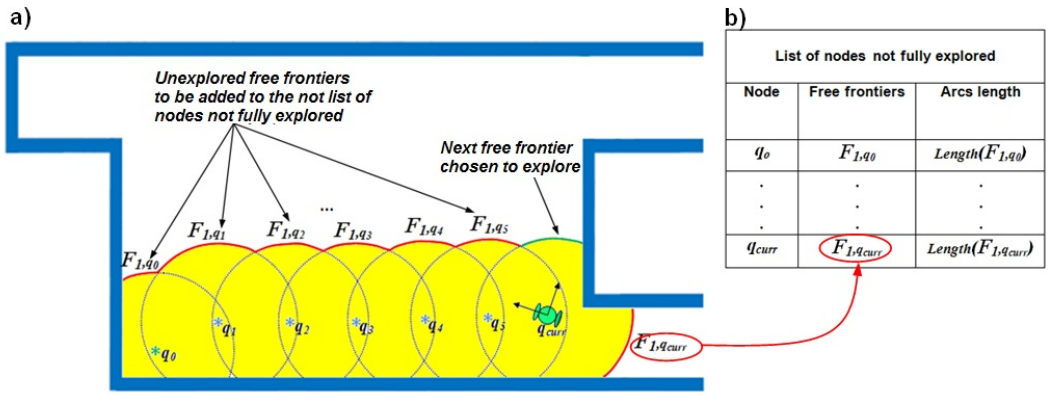


Fig. 8. Control de fronteras. a) El ambiente semiexplorado, los arcos $F_{i,qj}$ representan fronteras libres no exploradas, b) Lista de nodos no completamente explorados

introducido de control de fronteras (Figura 8) se realizará como sigue: una vez que la frontera aleatoria del nodo actual se elige, el control de fronteras agregará las fronteras inexploradas (Figura 8a) del nodo actual y su longitud del arco a una lista que será usada para planificar un camino a estos nodos (Figura 8b).

La planificación de caminos será ejecuta usando el método A* en forma bidireccional, planificando la ruta desde el nodo actual a los nodos en la lista y desde los nodos en la lista hacia el nodo actual, finalizando cuando exista un camino entre el nodo actual y cualquiera de los nodos en la lista.

3.1.1. Algoritmo del grafo aleatorio de exploración

Como en el SRT, en nuestro método el grafo representa el mapa de camino del área explorada y es gradualmente construido extendiendo la estructura hacia las fronteras seleccionadas aleatoriamente, de tal forma que la nueva configuración esté contenida en la LSR del nodo actual. Cada nodo del grafo consiste de una configuración libre de colisión *q* que el robot ha alcanzado junto con la descripción de la región de seguridad local *S* que rodea a *q* percibida por los sensores. Además, un arco entre dos nodos representa un camino libre de colisión. La Figura 9 muestra el algoritmo del método de exploración propuesto.

```
INTEGRATED_REG_EXPLORATION (qinit, kmax)

1.  Node_Ag = 0
2.  qcurr = qinit
3.  L_Nodes_Ex = Null
4.  Scurr ← PERCEPTION(qcurr)
5.  for k=1 to kmax
6.      Scurr ← CURR_INTERSECTION(G,Scurr,qcurr,L_Nodes_Ex)
7.      F ← FRONTIERS(Scurr)
8.      if F ≠ Null
9.          Node_Ag= Node_Ag+1
10.         (Frand,θrand) ← FRONT_RAND(F)
11.         F ← REMOVE(F , Frand)
12.         qdest ← DISPLACE(qcurr , θrand ,α , r)
13.         MOVE_TO(qcurr , qdest)
14.         Sdest ← PERCEPTION(qdest )
15.         Frand ← VERIFICATION( Frand , Sdest)
16.         F ← F ∪ Frand
17.         if F ≠ Null
18.             L_Nodes_Ex = L_Nodes_Ex ∪ Node_Ag
19.         end
20.         G ← ADD(G, Node_Ag , qcurr , S, F)
21.         qcurr = qdest
22.         Scurr = Sdest
23.     else
24.         (P, Ind_Node) ← FIND_PATH(qcurr, L_Nodes_Ex)
25.         for i = 1 to length(P)
26.             MOVE_TO( qcurr , P(i) )
27.             qcurr ← P(i)
28.         end
29.         L_Nodes_Ex ← REMOVE(L_Nodes_Ex , Ind_Node)
30.     end
31. end
32. Return ( G )
```

Fig. 9. Algoritmo REG

El nodo inicial del algoritmo se considera como el nodo de salida y de retorno. Este nodo contiene la siguiente información: la posición inicial q_{init} y la región de seguridad local (LSR). Aunque éste ha sido creado, el nodo aún no se agrega al grafo debido a que éste no debería ser tomado en cuenta cuando se busque por posibles conexiones con los nodos vecinos. Con esta información el ciclo que controla todo el proceso puede empezar. Se realiza un proceso de verificación sobre todos los nodos vecinos de la posición q_{curr} en cada k iteración con dos objetivos. El primer objetivo es verificar las secciones de fronteras libres de estos nodos que se encuentran dentro de la actual LSR. Los nodos que contienen fronteras intersectantes se actualizan al remover la frontera o segmento de frontera del nodo vecino y del nodo actual. El segundo objetivo, es identificar un camino seguro para viajar entre las fronteras que interceptan los nodos vecinos y el nodo actual. Si se encuentra un camino, un arco entre estos dos nodos se agregará a la estructura. Este proceso se realiza con la función **CURR_INTERSECTION.**

Una vez que se realiza la verificación y actualización de la estructura, la función **FRONTIERS** obtendrá las fronteras libres restantes F en la LSR. Si se encuentra al menos una frontera libre, la función **FRONT_RAND** elegirá aleatoriamente una de ellas y el punto medio de la longitud del arco de la frontera elegida será la nueva dirección aleatoria a visitar, siempre que la longitud del arco de esta frontera no exceda un cierto umbral proporcional a la distancia que la nueva LSR pueda cubrir. En el caso contrario, el segmento de frontera proporcional a la longitud del arco que puede ser cubierta empezando en el extremo inicial de la frontera se elegirá, tomando el punto medio del segmento como la nueva dirección a explorar. La frontera que se elige aleatoriamente se elimina entonces del grupo de fronteras libres obtenidas con la función **FRONTIERS,** ya que por ahora ya no es libre.

Al contrario del método clásico SRT, nuestra aproximación no requiere una función de validación para verificar que la nueva dirección elegida no está dentro de la LSR de otro nodo, debido a que el control de fronteras realizado remueve fronteras contenidas en otros nodos (función **INTERSECTION_ACT**).

Una vez que la dirección aleatoria se elige, la función **DISPLACEMENT** obtiene la nueva posición q_{dest} a ser visitada dando un paso de longitud $\alpha\ r$ en la dirección aleatoria elegida donde la constante r representa el radio de la LSR. Por otro lado, la constante $\alpha < 1$ asegura que esta estará dentro de la LSR y que podrá ser alcanzada a través de un camino contenido en ella. Con la nueva posición objetivo q_{dest} calculada la función **MOVE_TO** llevará al robot desde la posición actual hasta la posición objetivo.

Una vez que el robot ha alcanzado la posición objetivo, se realiza un nuevo proceso de percepción para estimar su espacio circundante S_{dest}. Con esta información, la función **VERIFIES** estima la porción de la frontera previa elegida por **FRONT_RAND** que ha sido cubierta. Si la frontera no ha sido cubierta al 100% la función regresará la porción restante de la frontera para explorarla y para incluirla en el grupo de fronteras F del nodo previo que debe ser explorado.

La información acerca de las fronteras libres restantes en el nodo previo lo clasificarán como un nodo con posibilidad de explorar sus fronteras libres, y se agregará su cabecera a la lista de nodos a ser explorados.

Cuando el nodo se agrega a la estructura del grafo, las curvas en el mapa serán extendidas con la nueva información usando la función **ADD** y el ciclo comenzará de nuevo usando la información q_{dest} y S_{dest} como q_{curr} y S_{curr}.

Típicamente, cuando el espacio donde se encuentra el robot ha sido completamente explorado, el algoritmo fallará en encontrar una frontera libre y la exploración continuará en cualquiera de los nodos almacenados en la lista del control de fronteras. La búsqueda para el siguiente nodo a explorar es realizada utilizando el algoritmo de búsqueda en grafos A* en forma bidireccional. El método terminará cuando la lista de nodos a explorar esté vacía y no existan más fronteras libres; en este caso, el robot regresará al nodo inicial desde donde inicio la exploración.

El empleo bidireccional del algoritmo A* extenderá el camino tanto de la posición inicial como de la posición deseada hacia la posición

final alcanzada por el lado opuesto, finalizando cuando ambos caminos estén en el mismo nodo.

Esta estrategia es usada simultáneamente en nuestro método, con todos los nodos contenidos en la lista y termina cuando un camino es encontrado. Este proceso es realizado con la función **FIND_PATH**. La razón de buscar rutas individuales desde la posición actual hacia todos los nodos con posibilidad de exploración en lugar de simplemente una distancia euclidiana hacia el nodo más cercano, es debido a la existencia de espacios inaccesibles y otras estructuras, la distancia al nodo podría parecer pequeña mientras que la distancia para alcanzarlo puede ser demasiado grande.

Una vez que la trayectoria *P* es obtenida, el método **MOVE_TO** llevará al robot desde el nodo actual hasta el nodo donde la exploración continuará. Finalmente el índice del nodo elegido es eliminado de la lista de nodos que pueden ser explorados. Con el nuevo nodo a explorar, el método continuará haciendo el mismo proceso descrito en el algoritmo hasta que ninguna frontera permanezca inexplorada.

3.2. SLAM topológico con *B-Splines*

Aunque las estrategias de SLAM basadas en el filtro de Kalman han mostrado buenos resultados, estos continúan presentando problemas tales como la inconsistencia y la reducción en la exactitud en la estimación del estado del sistema debido al efecto que las linearizaciones producen sobre las estimaciones del robot y del mapa. Además las implementaciones de este filtro tienen un costo computacional proporcional al número de elementos en el mapa, limitando su aplicación a mapas con un número reducido de componentes.

Aunque muchos trabajos han emergido para solucionar estos problemas, nosotros nos hemos concentrado en el desarrollo de una nueva estrategia topológica parcialmente basada en el uso de subregiones contenidas en cada nodo del método de exploración y en curvas *B-Spline* para modelar los obstáculos. La falta de una matriz de covarianza y el uso de regiones limitadas del ambiente permite al problema de SLAM ser fácilmente atacado en tiempo real incluso para ambientes de grandes dimensiones.

3.2.1. Administración de datos

En esta sección explicaremos la interpretación que será dada a los datos provenientes del sensor para obtener las *Splines* paramétricas que representan el mundo físico circundante al robot, el proceso de asociación de datos usado para establecer correspondencias entre las *Splines* detectadas y las *Splines* contenidas en el mapa.

a) Adquisición de las *B-Splines*

Cuando un robot obtiene un nuevo conjunto de medidas de su ambiente a través de su sistema de percepción, el mundo circundante es sólo un sistema de puntos sin significado, vinculados entre sí sólo por la lógica de la ordenación proporcionada por el sensor de barrido. Con estos datos brutos, el primer objetivo es el de identificar claramente los objetos asociados con las mediciones agrupadas en subconjuntos para obtener finalmente las curvas *B-Splines* que representan la porción de los objetos detectados tan cerca como sea posible.

Una vez que los subconjuntos se obtienen, las mediciones correspondientes a cada objeto detectado se aproximarán usando curvas *B-Splines* de grado 3. El ajustar los puntos de medida con *B-Splines*, ayuda a reducir el ruido contenido en ellas.

Finalmente con el objetivo de que el proceso de localización se pueda ejecutar efectivamente, la invariabilidad de la curva se debe asegurar. Así, cada curva discreta debería almacenarse tomando puntos equidistantes en ella con una distancia ε entre cada punto.

Una vez que las *B-Splines* se han obtenido y elegido, se pueden buscar características específicas contenidas en las curvas que serán de gran importancia en el proceso de localización. Esencialmente, dos tipos de características serán buscados en las curvas: puntos de inflexión y esquinas.

El proceso para obtener ambas características en la curva está basado en el espacio de curvatura a escalar CSS [9] el cual se usa para recuperar características geométricas invariantes.

b) Asociación de datos

El proceso de asociación de datos adoptado para nuestra propuesta de SPLAM topológico no

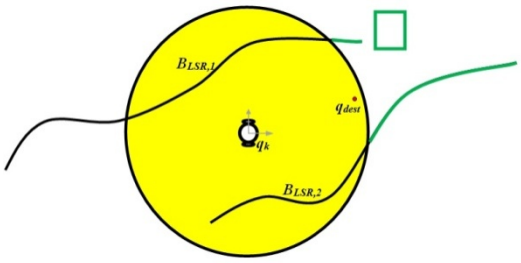


Fig. 10. LSR del robot en el instante *k*

se realiza considerando todos los objetos en el mapa construido hasta el instante *k*, sino sólo la porción del ambiente contenido en la LSR del último nodo construido en el método de exploración REG (Figura 10).

La asociación de *B-Splines* considerará inicialmente sólo los puntos del polígono de control que generan las curvas. En este paso, las distancias entre los puntos de los polígonos de control de todos los objetos (es decir, aquellos contenidos en la LSR actual y que son observados en la posición q_{k+s}) se obtienen asociando las curvas observadas con las curvas de referencia que se encuentren a una distancia menor de un cierto umbral (sección 2.3.1b).

Al final de esta primera etapa las *Splines* con un número mínimo μ_{min} de los puntos de control relacionados se asociarán y las restantes que no tienen ninguna relación con las ya establecidas, serán marcadas como nuevas curvas que podrían agregarse al mapa una vez que se corrija la posición del robot.

Si algunas curvas se han asociado en este punto, el siguiente paso consiste en buscar *esquinas* y *puntos de inflexión* contenidos en las curvas relacionadas. Si esto es posible, los elementos encontrados se usarán para realizar una asociación precisa entre cada par de curvas. En caso contrario, si ningún elemento se encontró (líneas o curvas demasiado suaves), el proceso de asociación fina se ejecutará en forma similar al que se describió en la sección 2.3.

Una vez que todos los elementos se han relacionado, se realiza una búsqueda de los puntos iniciales y finales de las curvas relacionadas. Para esto, se toman los puntos de inflexión o las esquinas más extremas contenidas en ambas curvas como puntos iniciales a partir

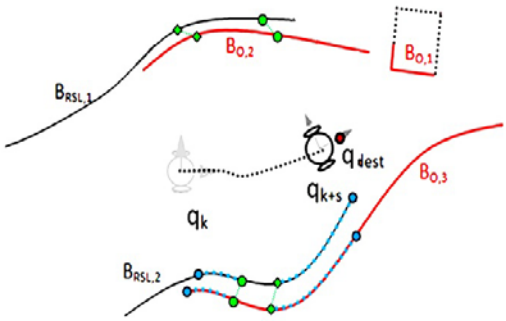


Fig. 11. Ejemplo de cómo se encuentran los puntos inicial y final de los segmentos de curva relacionados

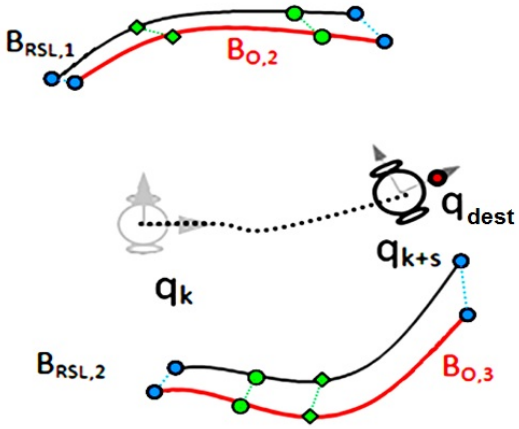


Fig. 12. Segmentos de curvas relacionados con el proceso descrito

de las cuales se usarán algunos puntos continuos sobre la curva paramétrica hacia su final. El número de puntos a considerar, es el número máximo de elementos que pueden tomarse en el segmento de la curva de menor longitud de las dos relacionadas desde los puntos característicos más extremos hacia el final de la curva. Este proceso se puede apreciar en la Figura 11.

Finalmente, cuando el proceso de asociación concluye, las curvas relacionadas tendrán una forma similar a la Figura 12.

3.2.2. Localización topológica con *B-Splines*

Hasta ahora, hemos hablado de las herramientas necesarias para la representación del ambiente así como las herramientas para la asociación de los datos. En esta sección,

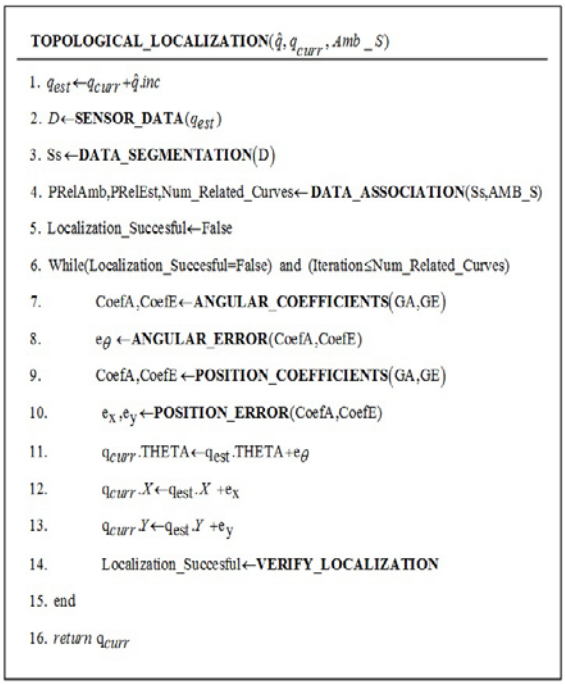


Fig. 13. Algoritmo de localización topológica

presentamos el método de localización topológica desarrollado, el cual forma parte de nuestra estrategia de SPLAM y cuyo algoritmo se muestra en la Figura 13.

Cada vez que el robot se mueve de una posición q_k a una posición q_{k+1} la estimación de la nueva posición del robot se obtiene agregando los incrementos en *x,* y, θ que se recolectaron de la información odométrica proporcionada por el robot en la última posición localizada.

Con la nueva posición estimada, se almacena una nueva percepción del ambiente una vez que se obtiene la información y esta se coloca espacialmente con respecto a esta posición (función **SENSOR_DATA**). Estas mediciones se segmentarán con la función **DATA_SEGMENTATION** para obtener subgrupos de mediciones pertenecientes a diferentes objetos.

El siguiente paso en el método, es encontrar la relación entre los objetos observados contenidos en los subgrupos encontrados y los objetos contenidos en la actual LSR. Esta tarea se realizará con la función **DATA_ASSOCIATION** y será el punto importante para realizar la corrección sobre la posición estimada y obtener la posición real del robot.

Una vez que los segmentos de la curva se han asociado y sus respectivos puntos inicial y final se han obtenido, esta información se usará para corregir el error en la posición estimada en dos etapas (primero angular y luego en traslación *x, y*) considerando lo siguiente:

$$\begin{aligned} X_{curr} &= \hat{X} + e_X \\ Y_{curr} &= \hat{Y} + e_Y \\ \theta_{curr} &= \hat{\theta} + e_\theta \end{aligned} \tag{6}$$

Donde $(x_{curr}, y_{curr}, \theta_{curr})$ es la posición actual del robot, $(\hat{x}, \hat{y}, \hat{\theta})$ es la posición estimada y (e_X, e_Y, e_θ) son los errores en *x, y,* θ. Así, la corrección angular se realiza tomando los segmentos de la curva asociados que pertenecen a la LSR (que servirá como referencia), en este caso los puntos iniciales y finales de cada una de ellas estarán unidos por segmentos de líneas, y además la función **ANGULAR_COEFFICIENTS** obtendrá un vector α_{ref} que contendrá los coeficientes angulares de cada una de estas curvas, el cual se calcula como sigue:

$$\alpha_{RSL,i} = \arctan \frac{p_{i,F} \sin(\varphi_{i,F} + \theta) - p_{i,I} \sin(\varphi_{i,I} + \theta)}{p_{i,F} \cos(\varphi_{i,F} + \theta) - p_{i,I} \cos(\varphi_{i,I} + \theta)} \tag{7}$$

Donde θ es el ángulo del robot en el instante cuando la actual LSR se obtuvo, φ es el ángulo del punto extremo (inicial y final) del segmento de curva relacionado con el marco de referencia del robot y *p* es la distancia desde el punto extremo (inicial y final) del segmento de curva al marco de referencia del robot. Al mismo tiempo, la función **ANGULAR_COEFFICIENTS** usa el mismo proceso en los segmentos de curva asociados que pertenecen a la última observación el cual claramente dará un vector diferente α_{curr} debido a la presencia de errores en la estimación. Una vez obtenido el coeficiente angular, la función **ANGULAR_ERROR** encontrará el error angular que se usará para corregir la posición angular (Figura 14). Para ello se utilizará la siguiente ecuación:

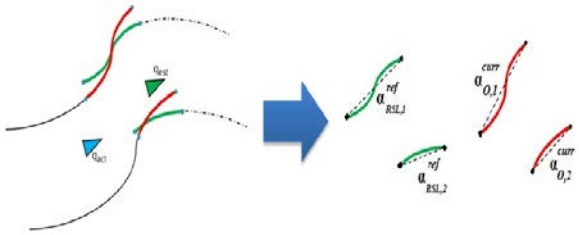


Fig. 14. Adquisición de los coeficientes angulares

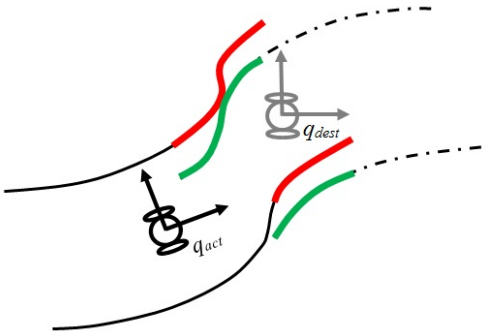


Fig. 15. Corrección angular realizada

$$e_\theta^* = \arg\min_{e_\theta} \sum_{i=1}^{Obj\_R} Ci(\alpha_i^{ref} - \alpha_i^{curr})^2 \quad (8)$$

La idea es entonces corregir la orientación estimada de tal forma que la norma de la diferencia entre los dos vectores sea mínima en un sentido de mínimos cuadrados. El peso C_i en la ecuación depende de la confiabilidad del par de características (líneas y curvas con puntos de inflexión o esquinas más confiables que las curvas suaves).

Después de obtener la corrección angular (Figura 15), entonces se procesan las correcciones en traslación. Esta corrección no requiere ningún tratamiento especial, la función **POSITION_COEFFICIENTS** sólo tomará las coordenadas de los puntos inicial y final de cada par de segmentos relacionados (X_{ref},Y_{ref}) y (X_{curr},Y_{curr}), ya que estos últimos dependen de e_x y e_y respectivamente. Entonces, la función **POSITION_ERROR** obtiene el mejor estimado de estas correcciones de traslación como sigue:

$$e_x^* = \arg\min_{e_x} \sum_{i=1}^{Obj\_R} Ci(X_i^{ref} - X_i^{curr})^2 \quad (9)$$

$$e_y^* = \arg\min_{e_y} \sum_{i=1}^{Obj\_R} Ci(Y_i^{ref} - Y_i^{curr})^2 \quad (10)$$

Los errores que se obtienen con estas ecuaciones se agregarán a la posición estimada para obtener la posición real del robot.

Finalmente, se realizará un de verificación con la función **VERIFY_LOCALIZATION**. Este asegurará que la localización ha sido exitosa, midiendo la distancia entre los puntos iniciales y finales de las curvas relacionadas y también entre las esquinas y los puntos de inflexión. Si estas distancias son menores que un cierto umbral, el proceso de localización será considerado como exitoso. En caso contrario, el proceso de localización fallará y este se reiniciará, pero esta vez considerando solamente curvas individuales en lugar del sistema completo.

3.2.3. Extensión del mapa

a) Agregando nuevos objetos al mapa

Esta forma de agregación no presenta un problema, cuando no se asocia una nueva observación con ninguna de las *Splines* en el mapa, esta curva será considerada como un nuevo elemento y las *Splines* que definen su geometría se agregan al mapa.

$$Map = \{O_1 \cup O_2 \cup \ldots \cup O_n \cup NO\} \quad (11)$$

Donde *Map* es el vector que contiene la información del mapa construido hasta el instante q_{curr}; O_j representa las curvas de tamaño variable que representan los obstáculos en el ambiente. Finalmente, *NO* representa al nuevo objeto que se intenta agregar, que como sabemos es un vector que contiene los puntos de control que forman la nueva curva.

a) Extensión de objetos al mapa

La técnica utilizada para la extensión de objetos contenida en el mapa se explicará utilizando la Figura 16.

Dadas dos curvas asociadas CA y CO (Figura 16a) cuya representación geométrica en los segmentos asociados son, iguales o al menos similares, el objetivo de este proceso es eliminar de alguna forma los puntos de control de la curva

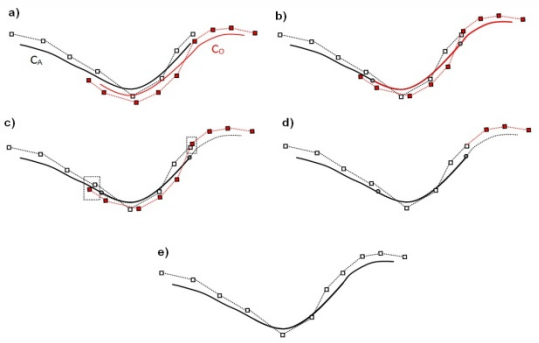


Fig. 16. Extensión de la *Spline* del mapa con la nueva información

observada, representando el área sobrepuesta entre las curvas. Para esto, una vez que la posición de las curvas se ha corregido a través del método de localización (Figura 16b), el proceso busca los puntos de control en la nueva curva que pertenecen a los puntos iniciales y finales de la asociación de los segmentos de curva (círculos grises en la Figura 16b) y cuya descripción ya está contenida en el mapa. Con estos puntos obtenidos (Figura 16c), el siguiente paso es eliminar estos dos puntos de control y los otros que están contenidos en el rango y entonces conectar los puntos de control de la curva que pertenecen al mapa con los puntos de control restantes de la nueva curva (Figura 16d).

4. Resultados experimentales

En esta sección hemos llevado a cabo numerosos experimentos con datos reales y simulados con la finalidad de verificar y validar los procedimientos y algoritmos propuestos en este trabajo de investigación. Los datos obtenidos con experimentos simulados han permitido verificar la exactitud y consistencia de las propiedades de los algoritmos comparándolos con otros existentes. Por otro lado, los experimentos en ambientes reales han permitido verificar la aplicabilidad y efectividad de estas técnicas.

4.1. Métodos de exploración

Como hemos mencionado repetidamente el control de movimientos (conocido como

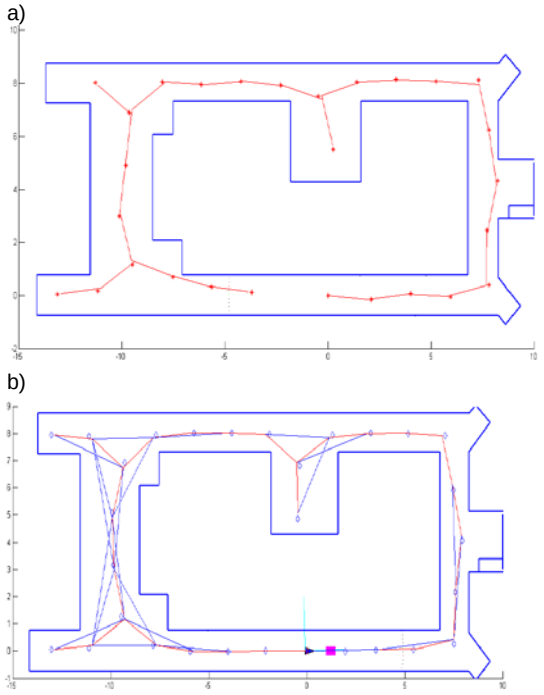
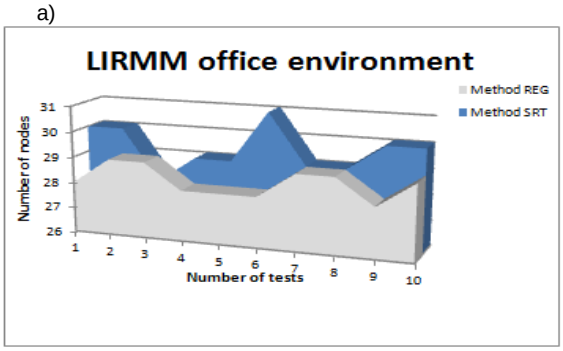


Fig. 17. Métodos de exploración SRT y REG. a) Estructura de datos generada por SRT. b) Estructura de datos generada por REG

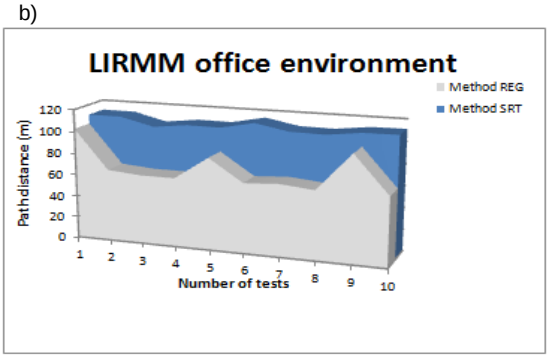
exploración en esta área) para la tarea de SPLAM, juega un rol muy importante en el rendimiento de la estrategia y en el tiempo de adquisición del mapa. A diferencia de la localización y la construcción del mapa necesarios en la tarea de SPLAM, la eficiencia del método de exploración puede analizarse y aprobarse individualmente. Por lo tanto, en esta sección se realizan pruebas comparativas entre el método de exploración SRT y el método de exploración propuesto REG desarrollado a partir de él.

Las pruebas a los métodos se realizaron usando un robot diferencial *Pioneer P3DX* equipado con un sensor *Hokuyo URG-04lx* el cual tiene un rango de detección de 0.02 a cuatro metros aproximadamente con una desviación atípica del 1% de la medida, una resolución angular de 0.36° y un ángulo de escaneo de 240°. Además, el robot tiene un anillo de 16 sensores ultrasónicos, de los cuales 6 de ellos están posicionados en la parte trasera y se usan

a)



b)



c)

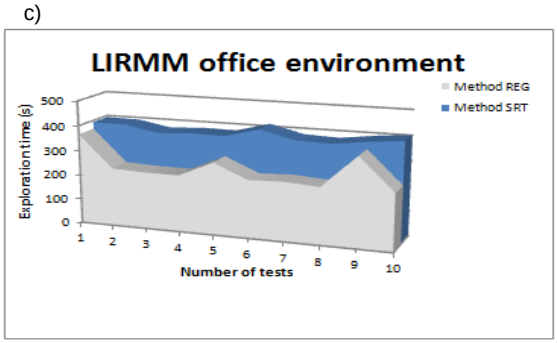


Fig. 18. Evaluación del rendimiento de los métodos de exploración. a) Nodos necesarios para cubrir el ambiente, b) Distancia recorrida para cubrir el ambiente, c) Tiempo necesario para la exploración del ambiente

para obtener información del ambiente en los 120° donde el sensor láser no puede ver.

El ambiente simulado usado para nuestros experimentos, es parte de las instalaciones del laboratorio de informática, robótica y micro electrónica de Montpellier (LIRMM) mostrado en la Figura 17.

Como hemos mencionado, tanto el método de exploración SRT presentado por Oriolo *et al.* [10] como el método REG desarrollado en este trabajo, generan una estructura que determina los caminos por los cuales el robot puede navegar. Es fácil observar en las Figuras 17a y 17b que el método REG produce una estructura de datos mucho más versátil que la que produce el método SRT.

Lo anterior puede ser verificado en la Figura 18 donde los datos obtenidos con los dos métodos son confrontados para verificar las suposiciones que hemos hecho. Los resultados se obtuvieron sobre una base de 10 pruebas.

Finalmente, basados en la información presentada en los gráficos anteriores, podemos concluir lo siguiente:

– Los nodos necesarios para la exploración (Figura 18a) en la mayoría de los casos es menor en el método REG que en el método SRT, esto se debe a que el método REG intenta obtener en cada nodo tanta información como sea posible. Por el contrario, el método SRT trabaja sobre la base de encontrar una dirección aleatoria sin importar la ganancia de la información.
– Otro elemento que debe ser considerado es la longitud del camino recorrido en la exploración del ambiente. En la Figura 18b podemos observar que la distancia recorrida en el método REG, es en la mayoría de los casos menor que la realizada con el método SRT. Lo anterior es debido a que la estructura de grafo permite usar atajos cuando el robot tiene que moverse dentro de la estructura.
– Otro aspecto importante que afecta el camino recorrido es el conocimiento de que los nodos que no han sido completamente explorados (gracias al concepto introducido del control de fronteras) es que el método REG sabe qué nodos deben ser revisitados. Esto, a diferencia de la metodología usada en SRT, que previene el hecho de que el robot tenga que revisitar nodos innecesarios sin posibilidad de exploración.
– Finalmente, en la Figura 18c observamos que en la mayoría de los casos el tiempo de ejecución para la exploración del ambiente es menor en el método REG que en el método SRT.

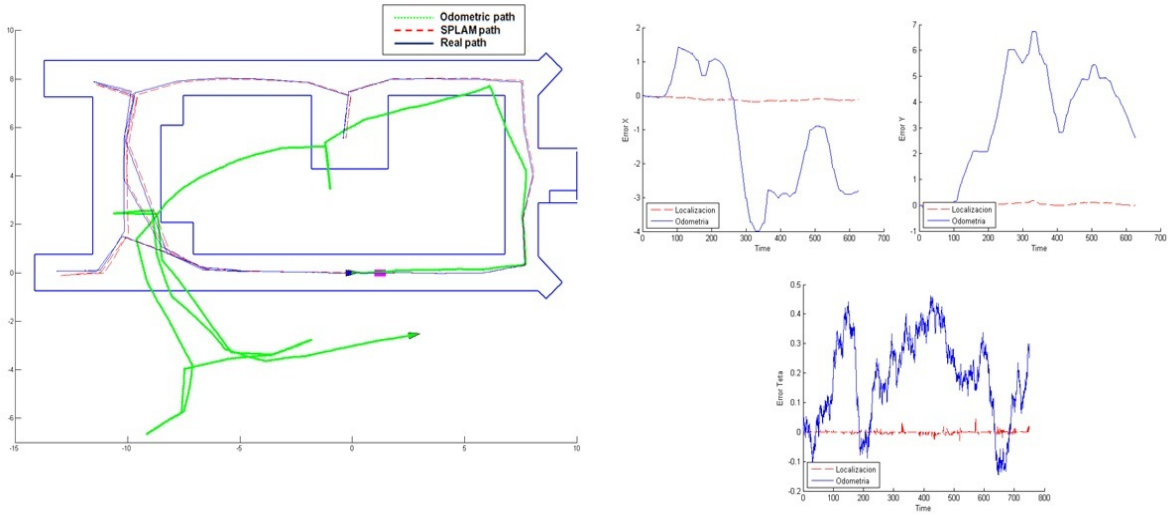


Fig. 19. Experimento de precisión y coherencia del método EKF-SPLAM clásico

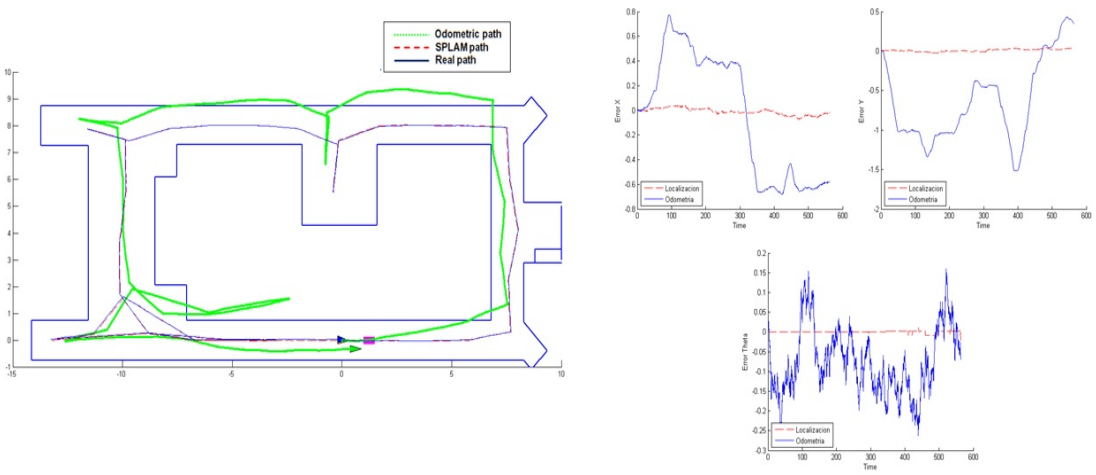


Fig. 20. Experimento de precisión y coherencia del método EKF-SPLAM Basado en *B-Splines*

4.2. Métodos de SLAM

4.2.1. Exactitud de los algoritmos

Como en la sección anterior, hemos experimentado con un ambiente simulado con la finalidad de evaluar la exactitud y efectividad de los algoritmos. Las Figuras 19, 20 y 21 muestran los resultados de los experimentos de SLAM para el caso del filtro de Kalman extendido clásico, para la estrategia de EKF basada en *B-Splines* y finalmente para la estrategia REG, la cual está basada en el uso de *B-Splines* para la representación de ambientes.

Las 3 imágenes del experimento, muestran en la parte izquierda, la trayectoria real del robot (azul continuo), la trayectoria odométrica (línea

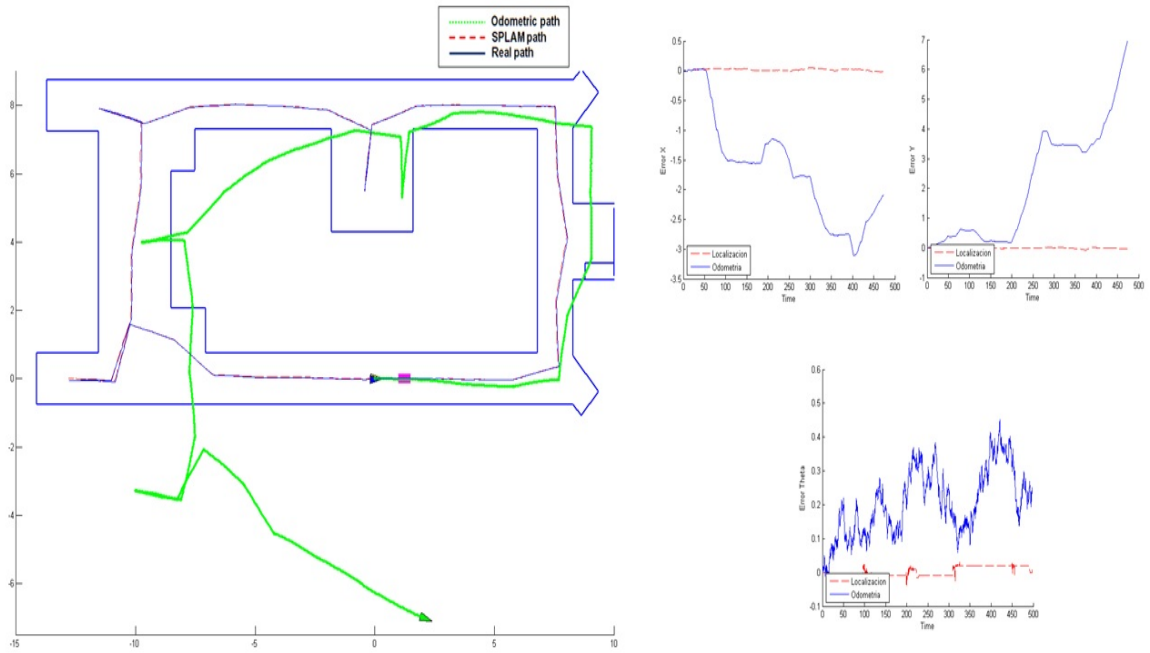


Fig. 21. Experimento de precisión y coherencia del método Topológico-SPLAM basado en *B-Splines*

verde punteada) y la trayectoria obtenida con el método de SPLAM (línea discontinua de color rojo). Por otro lado, cuando hay certeza sobre la trayectoria real del robot (como en la simulación), es posible realizar algunas comprobaciones para darnos una idea de la calidad de los algoritmos desde el punto de vista de su consistencia. Por esta razón, ha sido posible incluir en la mitad derecha de cada figura la representación del error odométrico (línea azul) en X, Y, θ, así como los errores de localización (línea roja discontinua).

De estos datos y tomando como referencia los errores mostrados por los métodos basados en el filtro de Kalman, concluimos que el método propuesto en este trabajo mantiene niveles de error similares y en algunos casos incluso mejores que los mostrados por otros métodos. (Figura 22).

4.3. Experimentos con datos reales

Finalmente, presentamos los experimentos con datos reales en ambientes reales para validar los resultados presentados. La Figura 23a muestra el ambiente real de oficinas usado para la prueba y la Figura 23b muestra el mapa obtenido mediante nuestra estrategia de SPLAM. Este ambiente fue construido utilizando 58 curvas *B-Splines* de grado 3 definidas por 1754 puntos de control. El tiempo requerido para la exploración del ambiente fue de 463 segundos.

Finalmente, considerando que nuestra estrategia ha sido creada para construir mapas de ambientes complejos, se realizó un último experimento usando el ambiente real mostrado en la Figura 24a. En este ejemplo, podemos observar que la construcción del mapa sería imposible usando un método clásico basado en líneas y puntos. La figura 24b muestra el mapa obtenido con nuestra propuesta.

5. Conclusiones

En este trabajo hemos presentado una nueva metodología para el proceso de exploración de ambientes complejos basada en la construcción de un grafo de exploración. Aunque el uso de

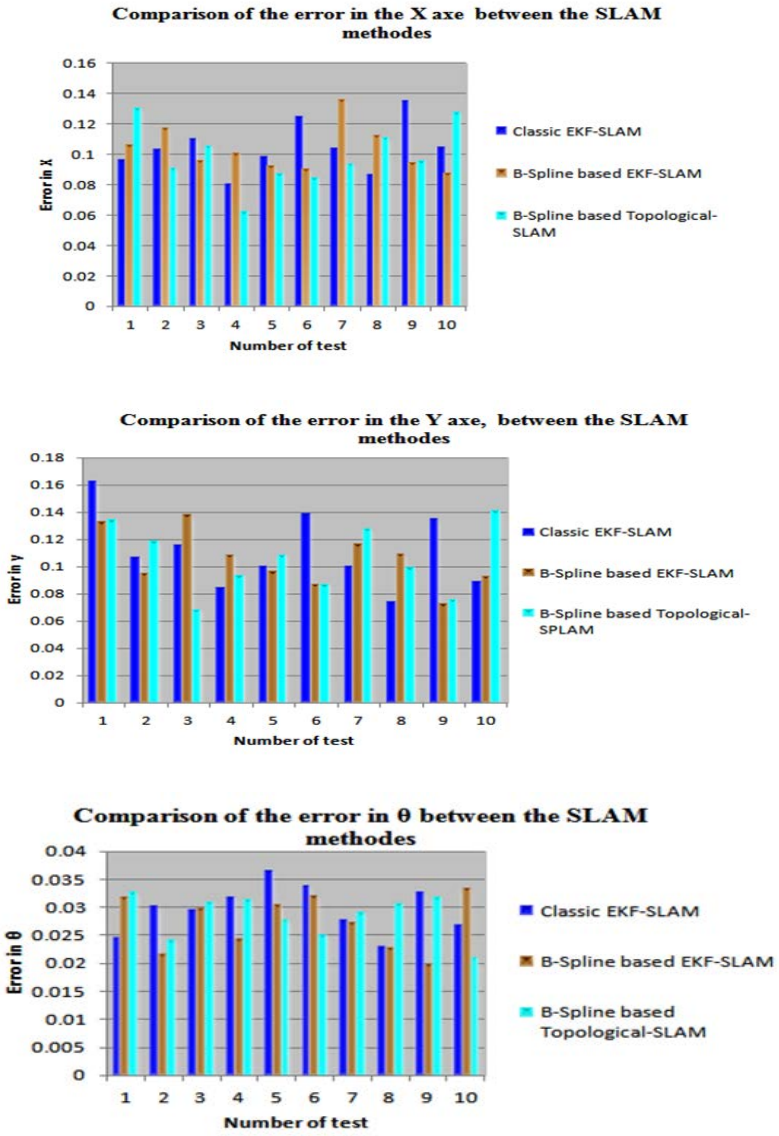


Fig. 22. Errores obtenidos con las estrategias de SPLAM

esta estructura ya ha sido usado para resolver este tipo de problemas, nuestra solución explota completamente la funcionalidad de la estructura para realizar navegaciones en la porción del ambiente conocido, independientemente de la finalidad. También, a diferencia de otras soluciones basadas en grafos, nuestra solución conserva la naturaleza aleatoria, ya que este tipo de soluciones han demostrado una mayor eficacia en el área de exploración de ambientes en los que no podemos tener una certeza de cuál será la siguiente mejor posición a ser explorada.

También, hemos desarrollado una estrategia de SLAM basada en curvas *B-Splines* para la representación de ambientes. Aunque este no es el primer trabajo donde ha sido usado este tipo

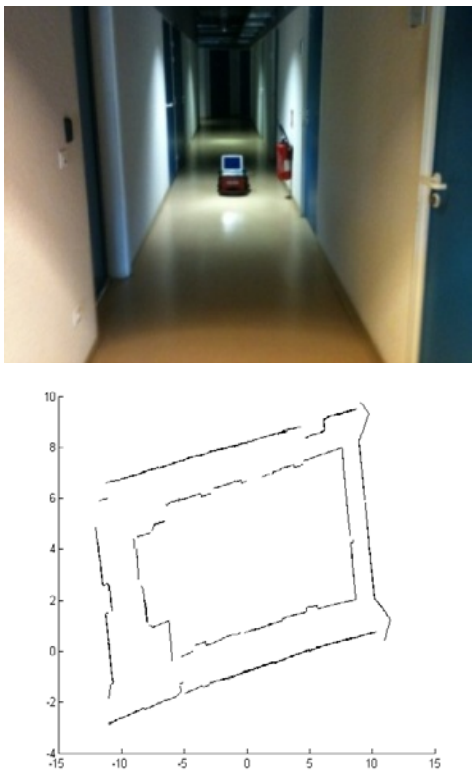


Fig. 23. Ambiente oficinas. a) Ambiente real usado para las pruebas. b) Ambiente adquirido con el método de SPLAM propuesto

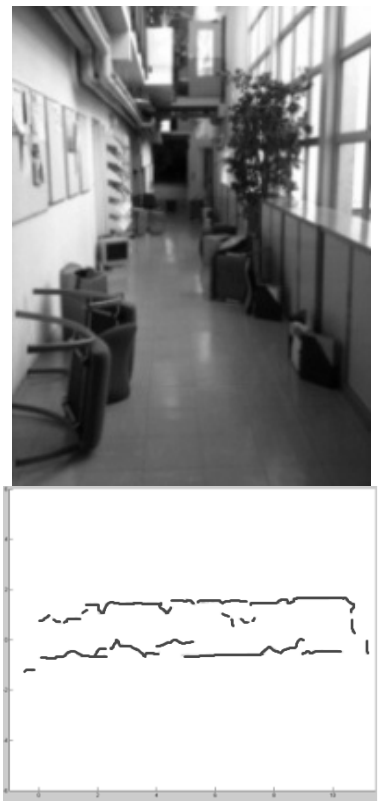


Fig. 24. Ambiente complejo. a) Ambiente real. b) Ambiente adquirido con el método de SPLAM propuesto

de representación, nosotros hemos adaptado algoritmos matemáticos del área de reconocimiento de patrones, con la finalidad de explotar la mayor cantidad de información y de esta manera proponer un método innovador de asociación de datos para el problema de SLAM.

Del mismo modo, la metodología aplicada para la corrección de la posición del robot y para el ajuste del ambiente está basada en información local usando la estructura del método de exploración, lo que representa una importante contribución ya que de esta forma el método no desperdicia tiempo y energía tratando de asociar datos fuera de rango. Así, nuestro método es apropiado para construir ambientes de grandes dimensiones.

Finalmente, a pesar de la aparente dificultad que la simbiosis de los métodos desarrollados podrían presentar, hemos logrado una cooperación armoniosa que fusiona las propiedades del método de exploración de ambientes con las propiedades del método de SLAM basado en curvas *B-Splines*.

Referencias

1. **Dietmayer, K. C. J, Sparbert, J., Streller, D. (2001).** Model based object classification and object tracking in traffic scenes from range images. *Proceedings of the IV IEEE Intelligent Vehicles Symposium*.
2. **Durrant-Whyte, H. & Bailey, T. (2006).** Simultaneous Localization and Mapping: Part I. *IEEE Robotics and Automation Magazine,* 13(2), 99–110.
3. **Espinoza, J.L, Sánchez, A.L., & Osorio, M. (2007).** Exploring unknown environments with

mobile robots using SRT-Radial. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS 2007), San Diego, CA, 2089–2094.

4. **Franchi, A., Freda, L., Oriolo, G., & Vendittelli, M. (2009).** The Sensor-based Random Graph Method for Cooperative Robot Exploration. *IEEE/ASME Transactions on Mechatronics*, 14(2), 163–175.
5. **González-Baños, H.H. & Latombe, J.C. (2002).** Navigation strategies for exploring indoor International. *Journal of Robotic Research*, 21(10-11), 829–848.
6. **Ishida, J. (1997).** The general B-spline interpolation method and its application to the modification of curves and surfaces. *Computer-Aided Design*, 29(11), 779–790.
7. **LaValle, S.M. (1998).** *Rapidly-exploring random trees: A new tool for path planning* (TR 98-11), Iowa, USA: Iowa State University.
8. **Makarenko, A.A., Williams, S.B., Bourgault, F., & Durrant-Whyte, H.F. (2002).** An experiment in integrated exploration. *IEEE/IRSJ International Conference on Intelligent Robots and Systems*, 1, Lausanne, Switzerland, 534–539.
9. **Mokhtarian, F. (1995).** Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5), 539–544.
10. **Oriolo, G., Vendittelli, M., Freda, L., & Troso, G. (2004).** The SRT Method: Randomized strategies for exploration. *IEEE International Conference on Robotics and Automation*, New Orleans, LA, USA, 5, 4688–4694.
11. **Pedraza, L., Dissanayake, G., Miro, J.V., Rodriguez-Losada, D., & Matia, F. (2007).** BS-SLAM: Shaping the World. *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.
12. **Pedraza, L., Rodriguez-Losada, D., Matia, F., Dissanayake, G., & Miro, J.V. (2009).** Extending the Limits of Feature-Based SLAM with *B-Splines*. *IEEE Transactions on Robotics*, 25(2), 353–366.
13. **Toriz, A., Zapata, R., Sánchez, A., & Osorio, M.A. (2010).** Integrated Exploration Based SRT-EKF. *2010 Ninth Mexican International Conference on Artificial Intelligence (MICAI 2010),* Pachuca, México, 171–176.
14. **Toriz, A., Sánchez, A., Zapata, R., & Osorio, M.A. (2010).** Building Feature-Based Maps with *B-Splines* for Integrated Exploration. *Advances in Artificial Intelligence (IBERAMIA 2010), Lecture Notes in Computer Science,* 6433, 562–571.
15. **Toriz, A., Sánchez, A., Zapata, R., & Osorio, A. (2011).** Mobile robot SPLAM for robust navigation, *Research in Computing Science*, 54, 295–305.

Alfredo Toriz Palacios recibió el grado de Maestro en Ciencias de la Computación de la Benemérita Universidad Autónoma de Puebla (BUAP) en 2008 y el grado de Doctor en Ciencias de la Computación y Robótica de la Universidad de Montpellier 2, Montpellier, Francia en 2012. Su área de interés incluye planificación de movimientos en robots, SLAM, robótica móvil, y animación por computadora.

Abraham Sánchez López recibió el grado de Maestro en Ciencias de la Computación de la Universidad de las Américas Puebla (UDLA) en 1996 y el grado de Doctor en Ciencias de la Computación y Robótica de la Universidad de Montpellier 2, Montpellier, Francia en 2003. Él es actualmente investigador en la Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación. Su área de interés incluye planificación de movimientos en robots, robótica móvil, animación por computadora y computación móvil.

Artículo recibido el 28/08/2012, aceptado el 21/06/2013.