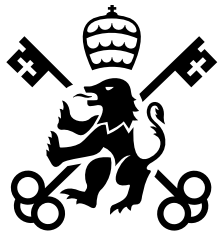


COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

ANÁLISIS COMPARATIVO DE ALGORITMOS DE LOCALIZACIÓN DE ROBOTS MÓVILES BASADOS EN FILTROS DE PARTÍCULAS

Autor: Julio Labora Gómez

Director: Jaime Boal Martín-Larrauri

MADRID

Agosto 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Análisis comparativo de algoritmos de localización de robots móviles basados en filtros de partículas

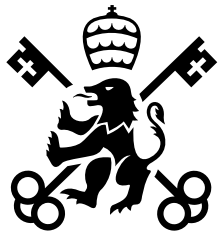
en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2019-2020 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Fecha: 26/ 08/ 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Fecha: / /



COMILLAS

UNIVERSIDAD PONTIFICIA

ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

ANÁLISIS COMPARATIVO DE ALGORITMOS DE LOCALIZACIÓN DE ROBOTS MÓVILES BASADOS EN FILTROS DE PARTÍCULAS

Autor: Julio Labora Gómez

Director: Jaime Boal Martín-Larrauri

MADRID

Agosto 2020

This dissertation was typeset with LaTeX and compiled in TeXmaker. The font families used are Bitstream Charter, Utopia, Bookman, Times New Roman and Computer Modern. Unless otherwise noted, all figures were created by the author using Microsoft PowerPoint®, Microsoft Excel®, Python® and MATLAB®.

ANÁLISIS COMPARATIVO DE ALGORITMOS DE LOCALIZACIÓN DE ROBOTS MÓVILES BASADOS EN FILTROS DE PARTÍCULAS

Autor: Labora Gómez, Julio.
Director: Boal Martín-Larrauri, Jaime
Entidad Colaboradora: ICAI - Universidad Pontificia Comillas

***Resumen*—La localización de robots móviles es clave en multitud de aplicaciones en la Industria 4.0, puesto que los robots móviles necesitan conocer su posición para poder realizar cualquiera de sus tareas. Un algoritmo muy popular para llevar a cabo la localización es el filtro de partículas, que puede ser modificado de diferentes formas para adaptarse a problemas específicos. El objetivo de este proyecto es comparar y analizar algunas de esas modificaciones en una aplicación real en un entorno simulado: el problema de localización global de un robot móvil 2-D en un laberinto. Las modificaciones que se van a comparar son 4 estrategias distintas de remuestreo: residual, estratificado, sistemático y remuestreo de la rueda. Sin embargo, debido al elevado coste computacional de las simulaciones, es necesario prefijar los parámetros de las simulaciones de estos 4 algoritmos. Para ello, el valor de cada parámetro se determina mediante un análisis comparativo previo en que se emplea otro algoritmo de remuestreo distinto, el remuestreo multinomial, para evitar sesgar las comparación. tres de los cuatro algoritmos comparados muestran resultados muy similares tanto en índice de éxito en la localización como en la precisión de localización. El cuarto, el remuestreo residual, da resultados considerablemente peores que los demás en esta aplicación específica.**

1 Introducción

La localización de robots móviles es un elemento esencial para el uso de robots en muchas aplicaciones industriales. Para realizar cualquier tarea, una de las primeras preguntas a las que debe responder cualquier robot móvil es *¿Dónde estoy?* Los algoritmos basados en filtros de partículas son algunas de las técnicas más comunes en la industria para determinar la localización del robot.

El filtro de partículas es un algoritmo recursivo no paramétrico que emplea un conjunto de muestras denominadas partículas para representar la distribución de un proceso estocástico. Este algoritmo proveed una representación continua del estado del sistema considerando varias hipótesis simultáneamente. Es, por tanto, un estimador de estado con una función multimodal.

El modelo espacio de estado puede ser no-lineal en una o varias de sus componentes y las distribuciones de los ruidos pueden ser cualesquiera (no necesariamente gaussianas). Este algoritmo es por ello muy versátil y se utiliza en multitud de campos.

En el campo de la robótica, existen múltiples problemas que pueden afrontarse con este algoritmo. Los más populares son: localización global, problema del robot secuestrado, *SLAM* (que significa *Simultaneous Localization and Mapping*, es decir, *Localización y Mapeo Simultáneos*) y localización en entornos dinámicos.

En el problema de localización global el robot es situado en un entorno conocido, pero su posición dentro de ese entorno le es desconocida. El problema del robot secuestrado es muy similar, con la salvedad de que la posición del robot puede ser modificada en cualquier momento sin ningún aviso [1] [2]. El *SLAM*, que es uno de los más populares en la industria, requiere que el robot navegue por el entorno y recoja información no solo acerca de su posición en él, sino también del propio entorno [3] [4]. La localización dinámica emplea algunos de los principios subyacentes del *SLAM*, con el inconveniente de que áreas que eran consideradas ya conocidas en el *SLAM* porque ya habían sido exploradas, ahora pueden haber sufrido algunos cambios.

2 Estado del arte

El filtro de partículas básico consta de 4 pasos:

- Inicialización
- Asignación de pesos y remuestreo
- Actualización del estado de las partículas
- Finalización

Durante la inicialización se seleccionan N partículas de una distribución uniforme en el espacio de las variables consideradas. Después, el peso de cada partícula se calcula empleando una cierta función de manera que sea proporcional a la probabilidad condicionada de que la partícula sea una representación precisa del estado del sistema. Posteriormente, las partículas se remuestrean en función de su peso. El estado de las partículas es actualizado teniendo en cuenta las dinámicas del sistema en el período de muestreo correspondiente. Los pasos 2 y 3 se repiten hasta que se cumple el criterio de finalización y la posición el robot se considera conocida. Este algoritmo es muy costoso computacionalmente, puesto que requiere replicar un proceso un elevado número de veces (la cantidad de partículas puede puede variar desde unos pocos miles hasta cientos de miles o incluso millones) [5].

Sin embargo, existen multitud de modificaciones sobre el algoritmo básico que pueden lidiar mejor con problemas específicos, como los mencionados en la Sección 1, y pueden compensar las desventajas del algoritmo convencional, como la degeneración de partículas, la elevada varianza, el elevado número de muestras necesario o el alto coste computacional.

Una de estas modificaciones consiste en modificar el número de muestras usado en el filtro de partículas en los distintos períodos de muestreo. Esto es clave para reducir el tiempo de simulación, puesto que el tiempo de computación crece, en general, de forma lineal con el número de partículas.

Una alternativa al algoritmo convencional es el uso de la suma de pesos de las partículas para determinar si el número de muestras que se han tomado es suficiente [6].

Otra modificación consiste en introducir nuevas partículas en cada iteración del algoritmo. Esto es una técnica que puede tratar con el problema del robot secuestrado. También es útil emplearla en el problema de la localización global para otorgar más robustez al algoritmo. Un inconveniente de esta técnica es que la constante inicialización de partículas aumenta el coste computacional del algoritmo [7].

Respecto del segundo paso, el cálculo de los pesos y remuestreo, se pueden emplear varias técnicas. El objetivo del remuestreo es reducir la degeneración de partículas, que es la concentración de la mayor parte del peso en un número reducido de muestras, de forma que los pesos del resto son despreciables. Algunas de las técnicas tradicionales más conocidas son: multinomial, estratificado, sistemático, residual y de la rueda [8] [9] [10]. Algunas variaciones más modernas igualmente populares incluyen remuestreo residual sistemático, modificado y distribuido [11] [12].

El remuestreo multinomial es quizá la técnica más intuitiva. Se generan N número aleatorios independientes en una distribución uniforme $(0, 1]$. Después se utiliza cada uno de ellos para seleccionar una partícula proporcionalmente a

Algorithm 1: Remuestreo de la rueda

```
index = U(1, N)
beta = 0
for i = 1 : N do
    beta = beta + U(0, 2 * max(weights))
    while weights[index] < beta do
        beta = beta - weights[index]
        index = index + 1
    end
    draw(particle[index])
end
```

su peso. Este método es muy sencillo pero con una varianza muy elevada (que favorece la degeneración de partículas) y extremadamente ineficiente computacionalmente, por ello en la práctica no se utiliza.

Para el remuestreo estratificado, el conjunto de partículas es dividido en varios subconjuntos (estratos), dividiendo el intervalo $(0, 1]$ en N subintervalos disjuntos y generando posteriormente un número aleatorio en cada uno de ellos. Las partículas se toman después de la misma forma que en el remuestreo multinomial. Esta técnica tiene menor varianza que la anterior y es más eficiente computacionalmente.

Una variación de esta técnica es el remuestreo sistemático, en el cual solamente un número aleatorio es generado $u_1 = (0, 1/N]$. El resto se calculan según la Ecuación 1.

$$u_n = u_1 + \frac{n-1}{N} \tag{1}$$

Para el remuestreo residual, todas las partículas con peso normalizado mayor que $1/N$ se replican un cierto número de veces, siendo este número $n_i = \lfloor N w_i \rfloor$ para la i-ésima partículas, cuyo peso normalizado es w_i. Después, $m = N - \sum n_i$ partículas son remuestreadas del conjunto de partículas inicial utilizando los residuos de los pesos en vez de los pesos originales. Estos residuos se calculan según la Ecuación 2. Para esta selección, se puede emplear cualquier método de remuestreo.

$$\hat{w_i} = w_i - \frac{n_i}{N} \tag{2}$$

Existen múltiples variaciones del remuestreo de la rueda. En esta sección se incluye el pseudocódigo de la que se emplea en este proyecto.

Una de las variaciones de los métodos de remuestreo es el remuestreo modificado. Esta técnica muestrea las partículas considerando funciones de los pe-

sos en vez de los propios pesos. Utilizando funciones es posible beneficiar a las partículas grandes o a las pequeñas según se desee acelerar la convergencia o aumentar la diversidad de partículas respectivamente. Esto se puede lograr ajustando el parámetro *alfa* en la Ecuación 3, en la que el peso es calculado como una función de la probabilidad asociada a la partícula.

$$\omega_i = (p_i)^{\alpha} \tag{3}$$

3 Metodología

En esta sección, se detalla la información del análisis comparativo: sus objetivos, los algoritmos comparados, la metodología y el entorno y los medios empleados.

El objetivo de este proyecto es compara múltiples variaciones del algoritmo del filtro de partículas y determinar si algunas son mejores que otras, cuál es la mejor para el problema de localización global y cuantificar las diferencias. Las variaciones comparadas son cuatro algoritmos de remuestreo: estratificado, sistemático, residual y de la rueda. Todos ellos se emplean en su versión *modificada*, puesto que el valor de *alfa* es distinto de 1.

Los algoritmos se evalúan en su capacidad de resolver un problema de localización global. Un robot móvil que opera en entornos 2-D es situado en un entorno simulado que le es conocido, pero sin conocer su localización en él. El robot intentará localizarse un número de veces, utilizando cada uno de los algoritmos para generar los datos necesarios para realizar la comparación. Cada intento se denominará expedición. Sin embargo, para poder realizar una expedición el robot necesita más que un filtro de partículas: requiere e la habilidad de procesar lecturas de sensores, un algoritmo de navegación para desplazarse por el entorno y otras componentes. Como estas no son objeto del análisis de este trabajo, se consideran dadas.

Antes de la comparación de los algoritmos de remuestreo se han realizado 3 tareas: reducción considerable del tiempo de computación de las simulaciones, automatización de las mismas para que no requieran intervención humana y optimización de algunos de los parámetros datos que se cree que tienen un mayor impacto en la comparación. En particular, la reducción del tiempo de computación ha sido crucial para los análisis, debido al elevado coste computacional del algoritmo del robot.

Aquellos parámetros que se han optimizado para el análisis comparativo son: *alfa* y el número de partículas. Además, múltiples estrategias para adaptar el número de partículas de forma dinámica también se han probado. Estos análisis se han llevado a cabo empleando el remuestreo multinomial para evitar sesgar la comparación en favor de uno de los algoritmos.

para comparar los algoritmos se consideran dos métricas clave: el índice de éxito (esto es, cuántas veces el robot se localiza correctamente) y la precisión en la localización. Una expedición se considera exitosa si el error de localización es menor de 0,65 metros en distancia y menor de 45 grados (0,78 radianes aproximadamente) en ángulo. Respecto al criterio de finalización, se ha empleado clusterización aglomerativa [13]. Dicho criterio consiste en una condición doble: el clúster de mayor tamaño, que es la mejor representación del estado del robot, debe contener más del 90% de las partículas existentes, mientras que el segundo clúster de mayor tamaño debe contener menos del 5%. En rigor, sería conveniente realizar un análisis de sensibilidad de estos parámetros, pero debido al tiempo de computación requerido no se ha llevado a cabo.

Debido a los problema de computación, el problema de localización global afrontado en este análisis es una versión simplificada del convencional. Como se puede ver en la Figura 1, todas las paredes del entorno son verticales u horizontales (cuando se representan en planta). Esto permite que la orientación inicial de las partículas pueda limitarse a cuatro valores: 0, 90, 180 y 270 grados; puesto que el robot podría rotar sobre sí mismo hasta una de esas posiciones sin ningún problema utilizando una pared como referencia. Esto reduce el tiempo de computación considerablemente, puesto que una de las variables se inicializa ahora de forma discreta y esto hace que se requieran solo una fracción de las partículas frente al caso canónico.

Para realizar las simulaciones la navegación, el filtro de partículas y el algoritmo de clusterización están programados en *Python 3.7*. Esta componente interactúa a través de una API con el simulador de robots *V-REP PRO EDU 3.6.2*. El robot empleado es el *Pioneer 3-DX*, un robot de dos ruedas con cinemática diferencial y 16 sensores sónar con un rango de 1 metro. Este tipo de robot es muy popular en la investigación, no solo en entornos simulados, sino también en entornos reales [14]. Todo esto se ha ejecutado en un sistema operativo *Windows 10*. El hardware utilizado es un ordenador portátil, modelo *Lenovo Yoga C930* con 16 GB de memoria RAM y un procesador de 8 núcleos modelo *i7-8550U @ 1.80 GHz*.

4 Resultados

Los parámetros que han sido analizados son: *alfa*, el número de partículas y la estrategia de adaptación del número de partículas. Respecto a la estrategia de adaptación, solo dos alternativas han sido comparadas: número de partículas constante y reducción constante del 5% de las partículas en cada iteración del filtro de partículas.

El parámetro *alfa* toma el valor de 1 en el algoritmo convencional. Por ello,

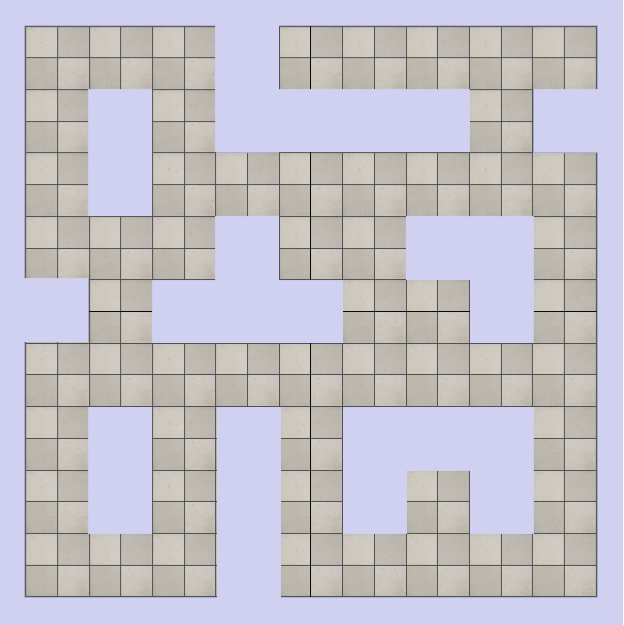


Figure 1: Entorno donde opera el robot. Mide 9 m x 9 m, y consiste de 81 casillas de 1 m x 1 m.

para seleccionar el valor óptimo, se ha comparado el rendimiento del algoritmo con valores entre 0,7 y 1,3 en saltos de 0,1. Es importante emplear valores cercanos a 1 porque el valor de *alfa* implica un compromiso entre velocidad de convergencia y éxito en la localización.

La Figura 2 muestra una tendencia ligeramente decreciente[1]. A pesar de lo reducido de la muestra, se puede observar que los valores 0,7 y 0,8 dan resultados notablemente mejores que los demás.

Considerando los resultados observados en la Figura 2, 0,7 y 0,8 parecen los mejores valores, siendo 0,7 ligeramente mejor. Sin embargo, para compensar el sesgo introducido por el hecho de que el remuestreo multinomial tiene una varianza mayor que los demás algoritmos, finalmente se elige el valor 0,8.

El número óptimo de partículas y la estrategia de adaptación se han determinado de manera similar, considerando el compromiso entre tiempo de computación e índice de éxito. Por cuestiones de brevedad, no se incluyen los resultados, sino solamente las elecciones finales (Tabla 1).

A continuación, los algoritmos de remuestreo mencionados se comparan: residual, estratificado, sistemático y de la rueda. Para generar los datos, se han realizado 500 expediciones con cada uno de ellos. Estas consisten en 5 expediciones

[1]Esta tendencia no es monótona debido al tamaño reducido de la muestra, pero teóricamente debería serlo.

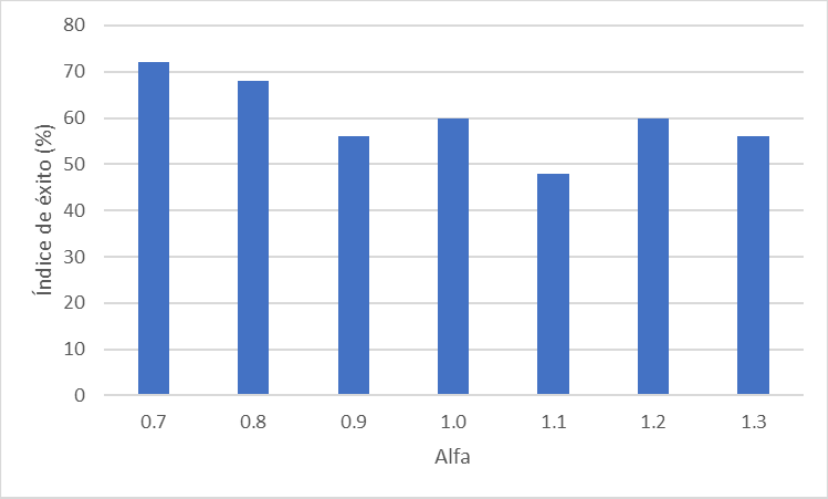


Figure 2: Índice de en la localización en función de *alfa*.

Table 1: Resumen de los parámetros.

Parámetro	Valor final
Alfa	0,8
Número de partículas	2500
Estrategia de adaptación	Decreciente 5%

Table 2: Índice de éxito de cada algoritmo de remuestreo.

Remuestreo	índice de éxito (%)
Residual	64,0
Estratificado	80,0
Sistemático	82,4
De la rueda	81,6

para 100 puntos de inicio distintos, elegidos aleatoriamente entre las localizaciones posibles del robot (considerando solo 4 valores de orientación, como se ha comentado en la Sección 3). El tiempo de simulación total para esta parte ha sido de aproximadamente 36 horas por técnica, es decir, cerca de 6 días.

Observando la Tabla 2, es muy notable que, mientras que tres de los algoritmos dan resultados muy similares, el cuarto es mucho peor. El remuestreo residual muestra un índice de éxito similar al del remuestreo multinomial, cerca del 65%, y bastante alejado de los demás, que superan el 80%.

La Figura 3 muestra como el remuestreo residual converge más rápido que los demás, y una vez más, los otros tres algoritmos presentan muy similares (casi idénticas de hecho). La rápida convergencia del algoritmo residual es teóricamente un punto a favor, mostrando el compromiso entre índice de éxito y velocidad. Sin embargo, siendo la media tan próxima a las demás, parece que en muchos casos el algoritmo de remuestreo residual converge prematuramente, y a pesar de haber dos o más localizaciones aun factibles elige una de ellas al azar, y son aquellos casos en los que falla que explican el menor índice de éxito.

Dicho esto, la diferencia en la velocidad de convergencia no es tan elevada, por lo que considerando los datos disponibles es difícil pensar en una aplicación industrial donde un incremento del 15% en la velocidad de convergencia justifique una pérdida de 15 puntos porcentuales de índice de éxito.

La distribución de los errores de localización, tanto en distancia como en ángulo, han mostrado igualmente muy poca diferencia entre las tres mejores técnicas. Teniendo en cuenta todos los datos, no es posible asegurar que uno de los algoritmos sea el mejor. De ser necesario decantarse por uno, este sería el remuestreo sistemático, puesto que tiene un índice de éxito ligeramente superior al de los otros, sobre todo teniendo en cuenta que algunos de los parámetros prefijados están sesgados para favorecer al remuestreo de la rueda.

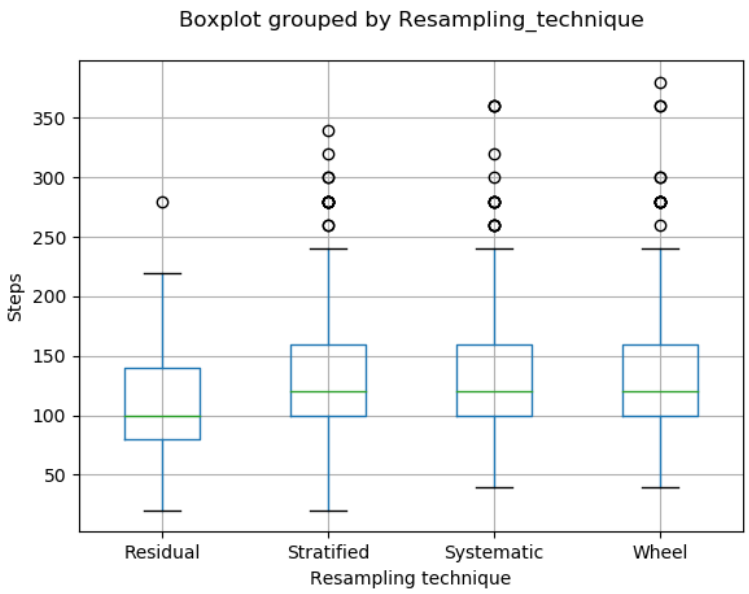


Figure 3: Número de iteraciones antes de la localización.

5 Conclusiones

Tras la comparación, no hay un claro ganador. El remuestreo residual es claramente pero que los demás, pero estos están muy próximos entre ellos en todas las métricas abordadas. El remuestreo sistemático es posiblemente el mejor por poca diferencia, pero es necesario profundizar en el análisis para determinarlo con seguridad. Lo cierto es que las tres técnicas mencionadas muestran resultados muy parecidos. Son tan parecidos, que lleva a pensar que la razón de que los índices de éxito no sean más elevados no se debe al remuestreo, sino a las otras componentes del algoritmo del robot. Por ello, parece que mientras se use una de estas variantes el remuestreo muy probablemente va a ser una limitación al algoritmo.

Todo esto es un recordatorio de que, cuando se abordan problemas reales, hay muchas partes móviles que juegan un rol en los resultados, y no todas ellas se pueden controlar. Debido a esto, es importante considerar cuidadosamente la fiabilidad y universalidad de las conclusiones derivadas de este tipo de experimentos.

Además, es bastante notable el impacto que puede tener en los resultados la realización de ajustes menores en parámetros como *alfa*. Esto sugiere que, con tiempo y recursos suficientes para ajustar todos los parámetros presentes en este problema, la mejora resultante podría ser muy considerable.

Un futuro desarrollo en lo referente a estos análisis es la implementación exitosa de una estrategia de adaptación del número de partículas más sofisticada.

Esta característica es muy prometedora, puesto que permitiría reducir el número de partículas promedio, y por tanto reducir la computación, sin sacrificar índice de éxito.

Otro desarrollo prometedor, alineado con las tendencias de la industria, sería la transformación del algoritmo para poder realizar *SLAM* y replicar el análisis comparativo. Los resultados podrían cambiar radicalmente, solo debido a las diferentes características del problema.

Bibliografía

[1] I. Bukhori and Z. H. 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, no. 4, p. 17298814-1771746, 2017.

[2] I. Bukhori, Z. H. Ismail, and T. Namerikawa, "Detection strategy for kidnapped robot problem in landmark-based map monte carlo localization," in 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), pp. 75–80, IEEE, 18/10/2015 - 20/10/2015

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] M. Lin, C. Yang, D. Li, and G. Zhou, "Intelligent filter-based slam for mobile robots with improved localization performance," IEEE Access, vol. 7, pp. 113284–113297, 2019.

[5] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Intelligent robotics and autonomous agents, Cambridge Mass.: MIT Press, 2005.

[6] D. Fox, "Adapting the sample size in particle filters through kld-sampling," The International Journal of Robotics Research, vol. 22, no. 12, pp. 985–1003, 2003.

[7] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), pp. 1225–1232, IEEE, 24-28 April 2000.

[8] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," IEEE Signal Processing Magazine, vol. 32, no. 3, pp. 70–86, 2015.

[9] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in 2006 IEEE Nonlinear Statistical Signal Processing Workshop, pp. 79–82, IEEE, 13/09/2006 - 15/09/2006.

[10] W. Adiprawita, A. Suwandi Ahmad, J. Sembiring, and B. R. Trilaksono, "A novel resampling method for particle filter for mobile robot localization," International Journal on Electrical Engineering and Informatics, vol. 3, no. 2, pp. 165–177, 2011.

[11] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," IEEE Transactions on Signal Processing, vol. 53, no. 7, pp. 2442– 2450, 2005.

[12] B. Balasingam, M. Bolic, P. M. Djuric, and J. Miguez, "Efficient distributed resampling for particle filters," in Efficient distributed resampling for particle filters, pp. 3772–3775, IEEE, 22/05/2011 - 27/05/2011.

[13] K. Sasirekha, Prassanna Baby, "Agglomerative Hierarchical Clustering Algorithm- A Review," 2013

[14] S. I. Martínez, J. A. C. Rocha, J. L. Menchaca, M. G. T. Berrones, J. G. Obando, J. P. Cobos, and E. C. Rocha, "An autonomous navigation methodology for a pioneer 3dx robot," Computer Technology and Application, vol. 5, no. 2, 2014.

COMPARATIVE ANALYSES OF MOBILE ROBOTS LOCALIZATION ALGORITHMS BASED ON PARTICLE FILTERS

Abstract— **Mobile robot localization is key in multiple applications across Industry 4.0, as mobile robots need to know their location to be able to perform almost any task. One popular algorithm to perform the localization is the particle filter, which can be modified in several ways to approach different problems. The goal of this project is to compare and analyze some of those modifications in a real application in a simulated environment: global localization of a 2-D mobile robot in a maze. The modifications that are compared are 4 different strategies for particle resampling: residual, stratified, systematic and wheel resampling. However, due to the high computational cost of the simulations, the parameters of the simulations of these 4 algorithms need to be prefixed. For this purpose, the value for each parameter is determined performing a prior comparative analysis of the parameters using another resampling technique, multinomial resampling, to avoid biasing the comparison. Three of the four algorithms show very similar results both in success rate in localization and in the localization accuracy. The fourth one, the residual resampling, is shown to be considerably worse for this specific application.**

1 Introduction

The localization of mobile robots is an essential element for the use of robots in many industrial applications. In order to perform any task, one of the firsts questions any mobile robot should answer is *Where am I?* Particle filter based algorithms are some of the most common techniques used across all industries to determine the location of the robot.

A particle filter is a recursive non-parametric mathematical algorithm that uses a set of samples called particles to represent the distribution of an stochastic pro-

cess. This algorithm provides a continuous representation of the state of the system considering several hypotheses at the same time. It is, therefore, a state estimator with a multimodal function.

The state space model can be non-linear in one or several of its components and noise distributions can be of any kind (not necessarily Gaussian). This algorithm is therefore very versatile and can be used in many fields.

In the field of robotics, there are various problems that can be tackled with this algorithm. The most popular ones are: global localization, kidnapped robot problem, SLAM (that stands for *Simultaneous Localization and Mapping*) and localization in dynamic environments.

In global localization a robot is located in a known environment but its position within the environment is unknown. The kidnapped robot problem is quite similar but with the difference that the robots position can be changed at any time without notice [1] [2]. SLAM, which is one of the most popular ones in the industry, requires that the robot navigates in the environment and gathers information regarding not only its position within it, but about the environment itself [3] [4]. Localization in dynamic environments uses some of the underlying principles in SLAM, with the downside that areas that were considered already known in SLAM because they had already being explored, now can be slightly different.

2 State of the art

The basic particle filter algorithm has 4 steps:

- Particle initialization
- Weight calculation and resampling
- Particles update
- Localization criterion

During initialization N particles are sampled from a uniform distribution in the feasible state space. Then the weight of each particle is calculated using a function that is somehow proportional to the conditioned probability of that particle being an accurate representation of the system state. Afterwards, particles are resampled according to their weight. The state of the particles is updated taking into account the dynamics of the system in the corresponding sampling time. Steps 2 and 3 are repeated a number of times until the localization criterion is met and the robot location is considered known. This algorithm is computationally costly, because it requires the replication of a process a high number of times (number

of particles used can vary between a few thousands and hundreds of thousands or even millions) [5].

However, several modifications of the basic algorithm exist, so that it can better handle specific problems such as those mentioned in Section 1, and can also compensate the downsides of the conventional algorithm such as particle degeneracy, high variance, high number of samples required and high computational cost.

One of these modifications consists of changing de number of samples used in the particle filter across the sampling periods. This is key to reducing the simulation time as the computation cost increases linearly with the number of particles. There are several techniques to achieve this.

An alternative implementation is to use the sum of the weights of the particles to determine if enough samples had been drawn of more are needed [6].

Another modification is to introduce new particles in each iteration of the algorithm. This is a technique that can handle the kidnapped robot problem on itself, provided that enough particles are added. Nevertheless, its use in the global localization problem (even if the robot cannot be arbitrarily moved) is also useful to make the algorithm more robust. One downside is that the constant initialization of new particles increases the computation cost of the algorithm [7].

Regarding the second step, weight calculation and resampling, several techniques can be used. The goal of resampling is to minimize particle degeneracy, which is the concentration of most of the weight in a few number of particles, being, therefore, the weights of the rest of the particles negligible. Some of the most well-known traditional resampling tecniques are: multinomial, stratified, systematic, residual and wheel resampling [8] [9] [10]. Popular modern variations include residual systematic, modified and distributed resampling [11] [12].

Multinomial resampling is perhaps the most intuitive technique, as it is completely naive. N independent random numbers are sampled from a uniform distribution in the range $(0, 1]$. Then, each of them is used to draw a particle proportionally to its weight. This technique has a high variance, with increases particle degeneracy and is computationally inefficient, thus it is not used in practice.

For the stratified resampling, the set of particles is divided into several subsets (strata), dividing the $(0, 1]$ range in N disjoint subranges and generating afterwards a random number in each of them. Particles are then drawn in the same way as multinomial resampling. This technique has less variance than the previous one and is more computationally efficient.

A variation of this technique is systematic resampling, in which only one random number is generated $u_1 = (0, 1/N]$. The rest are calculated by the Equation 1.

Algorithm 1: Wheel resampling

```
index = U(1, N)
beta = 0
for i = 1 : N do
    beta = beta + U(0, 2 * max(weights))
    while weights[index] < beta do
        beta = beta - weights[index]
        index = index + 1
    end
    draw(particle[index])
end
```

$$u_n = u_1 + \frac{n-1}{N} \tag{1}$$

For residual resampling, all particles with normalized weight greater than $1/N$ are replicated a number of times, this number being $n_i = \lfloor N w_i \rfloor$ for the i-th particle, which has a normalized weight of w_i. Then, $m = N - \sum n_i$ particles are resampled from the initial particles set using the residuals of the weights instead of the original weights. These residuals are calculated by the Equation 2. For this selection, any other resampling method can be used.

$$\hat{w_i} = w_i - \frac{n_i}{N} \tag{2}$$

Several variations of the wheel resampling method exist. In this section, the pseudocode of the one used in this project is included.

One of the variations of the resampling methods is modified resampling. This technique resamples particles considering not their weights but functions of them. Using functions is possible to benefit smaller or bigger particles. This way, it is possible to adjust the functions to increase the particle diversity by benefiting smaller particles or to speed up convergence by benefiting bigger particles. This can be achieved tuning the parameter *alpha* in the Equation 3, in which the weight is calculated as a function of the probability associated to that particle.

$$\omega_i = (p_i)^{\alpha} \tag{3}$$

3 Methodology

In this section, the comparative analysis performed is detailed: its goals, the algorithms being compared, the methodology and the environment and means.

The goal of this project is to compare multiple variations of the particle filter algorithm and determine if some of them are better, which is the best one for the global localization problem and to quantify the differences. The variations compared are four resampling algorithm techniques: stratified, systematic, residual and wheel resampling. All of them are used in their *modified* version, as the value of *alpha* is different from 1.

The algorithms are evaluated in their ability to solve a global localization problem. A 2-D mobile robot is placed in a known, simulated environment but it does not know its location. It must gather information to determine it. The robot will try to localize itself several times using each of the algorithms to generate the data for the comparison. Each of this tries, is referred to as *expedition*. However, to be able to perform an expedition, the robot needs more than a particle filter: it requires the ability to process sensor readings, a navigation algorithm to move within the environment and other components. As these components are not the subject of this analysis, they are considered given.

Prior to the comparison of the resampling algorithms three tasks have been performed: considerable reduction of the required computation time of the simulations, automation of the simulations so that they do not require human intervention and fine-tuning of some of those given parameters which are believed to have the most impact in the comparison. In particular, the reduction of the computation time has been crucial for the analyses, because of the high computation cost of the particle filter.

The furtherly fine-tuned parameters are: *alpha* and the number of particles. Additionally, multiple strategies to adapt de number of particles have also been tested. This comparative analysis has been carried out using the multinomial resampling algorithm to avoid introducing bias in the analysis.

To compare the algorithms, two key metrics are considered: success rate (i.e. how many times the robot localizes successfully) and localization accuracy (i.e. when the robot localizes itself, how much is the error between the believed pose and the true pose). An expedition is considered successful if the localization error is lower than 0.65 meters (in euclidean distance) and lower than 45 degrees in orientation (0.78 radians approximately). Regarding the localization criterion, agglomerative clustering is used [13]. The finalization criterion consists of double condition: the biggest cluster, which is the best representation of the robot pose, must contain more than 90% of the existing particles, where as the second biggest cluster must contain less than 5%. Rigorously, a sensitivity analysis of each of these parameters should have been carried out, but this would have required an excessive amount of computation time.

As the computation time of the simulations is an issue in these analyses, the global localization problem that is being tackled is a simplified version of the regular problem. As can be seen in Figure 1, all the walls of the environment are

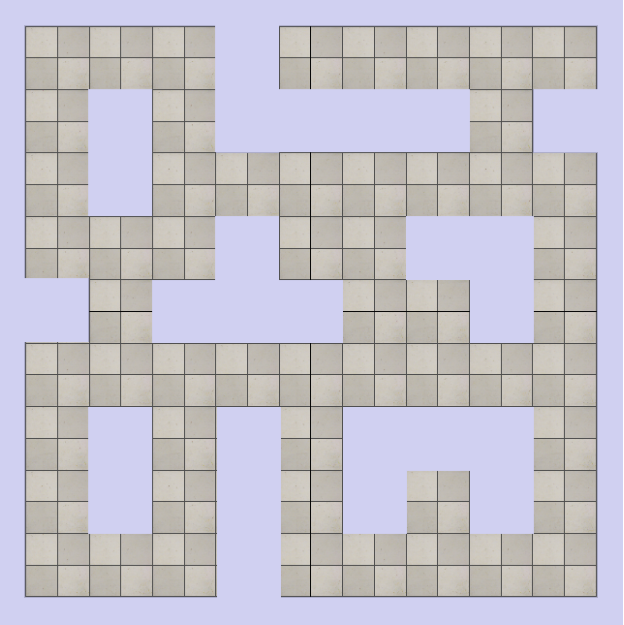


Figure 1: Robot environment. It is 9 m x 9 m, consisting of 81 1 m x 1 m cells.

either horizontal or vertical. This allows that the robot's initial orientation, and therefore the particles' initial orientation, can be limited to 4 values: 0, 90, 180 and 270 degrees, as the robot could be able to rotate to one of this positions anyway just by using a wall as a reference. This reduces the computation considerably, as one of the variables is now discrete in the initialization, and only a fraction of the particles is needed.

To perform the simulations the navigation, particle filter and clustering algorithms are programmed in *Python 3.7*. This component interacts through an API with the robot simulator *V-REP PRO EDU 3.6.2*. The robot being used is the *Pioneer 3-DX*, which is a two-wheeled differential-drive robot that has 16 sonar sensors with a range of 1 meter. This type of robot is quite popular in the research field, not only on simulated environments but also real ones [14]. All of this, is run on a *Windows 10* operating system. The hardware used is a laptop, model *Lenovo Yoga C930* with 16 GB of RAM and an 8-core *i7-8550U @ 1.80 GHz* processor.

4 Results

The parameters that have been analyzed are: *alpha*, the particle number and the particle number adaption strategy. Regarding the particle number adaption strategy, only two alternatives have been compared: constant number of particles and

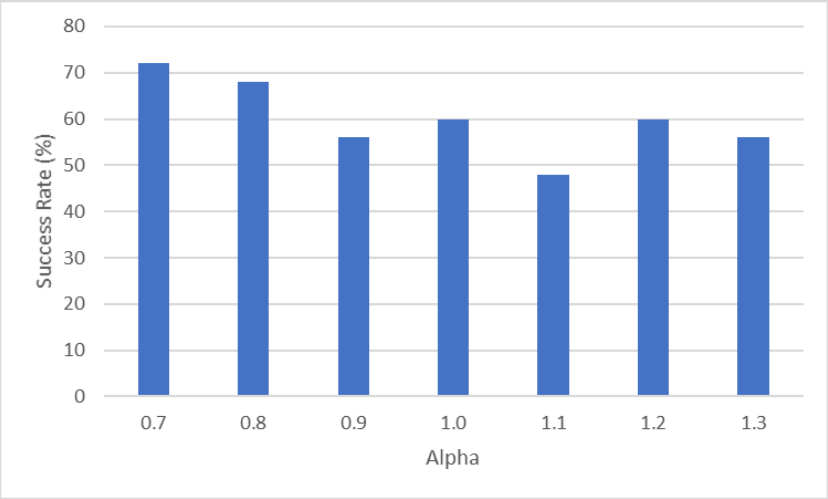


Figure 2: Localization success rate grouped by alpha.

a constant 5% decrease each iteration of the particle filter. .

The *alpha* parameter takes a default value of 1 in the conventional algorithm. Therefore, to select the optimal value, simulations were run in identical conditions varying only this parameter between 0.7 and 1.3 in steps of 0.1. Values close to 1 are used because regarding the value of *alpha* there is a trade-off between speed of convergence and localization success rate and therefore it cannot be highly increased or decreased.

The Figure 2 shows a slightly decreasing trend[1]. Despite the reduced size of the sample, it can be observed that the lower values of alpha 0.7 and 0.8 achieve higher success rates than the rest, almost 10 porcentual points more.

Taking into account the results shown Figure 2, 0.7 and 0.8 appear to be the best values, being 0.7 slightly better. However, to compensate for the bias introduced by the fact that multinomial resampling has a higher variance that the other algorithms, the final choice is *alpha* equal to 0.8.

The optimal number of particles and particle number adaption strategy had been determined using a similar approach, considering also the trade-off between computation time and successful localization. For the sake of brevity, the results are not included and only the final choices are (Table 1).

The aforementioned resampling algorithms are compared: residual, stratified, systematic and wheel resampling. To generate the data, 500 expeditions have been simulated for each of them. These 500 expeditions consist of 5 expeditions repeated for 100 different starting points, chosen at random within the feasible locations of the robot in the map (considering only 4 orientation values as ex-

[1]This trend is not monotonous mainly due to the reduced size of the sample.

Table 1: Parameter summary.

Parameter	Final value
Alpha	0.8
Particle number	2500
Adaption strategy	Decreasing 5%

Table 2: Success rate of each resampling algorithm.

Resampling	Success Rate (%)
Residual	64.0
Stratified	80.0
Systematic	82.4
Wheel	81.6

plained in Section 3). The total simulation time has been approximately 36 hours per resampling algorithm, about 6 days in total.

Observing the Table 2, it is quite noticeable that whereas three of the algorithms look quite similar, the fourth one, is much worse. Residual resampling shows a success rate similar to the one of multinomial resampling, close to 65%, and far from the over 80% of the other three.

Figure 3 shows that residual resampling converges the fastest, and once again, the other three are quite similar (almost identical in fact). The fast convergence of the residual algorithm is theoretically a positive feature, showing the trade-off between convergence speed and success rate. However, because the average is so close to the others, it seems that in many cases the residual algorithm converges prematurely and despite there being two or more possible locations for the robot it choses one of them at random, and are those cases in which that choice is wrong that lead to the lower success rate.

That being said, the difference in speed of convergence is not that high, so considering the data is hard to think of an industrial application in which a 15% increase in speed convergence is preferred over a difference of 15 porcentual points in success rate.

The distribution of the localization errors (both in distance and angle) have shown also very little difference between the three best techniques regarding the localization accuracy. Considering all these data, it is not possible to assure that one of the algorithms is the best. It seems it might be systematic resampling, as it has a slightly higher success rate than the others, specially taking into account that some of the preset parameters are biased in favor of the wheel resampling

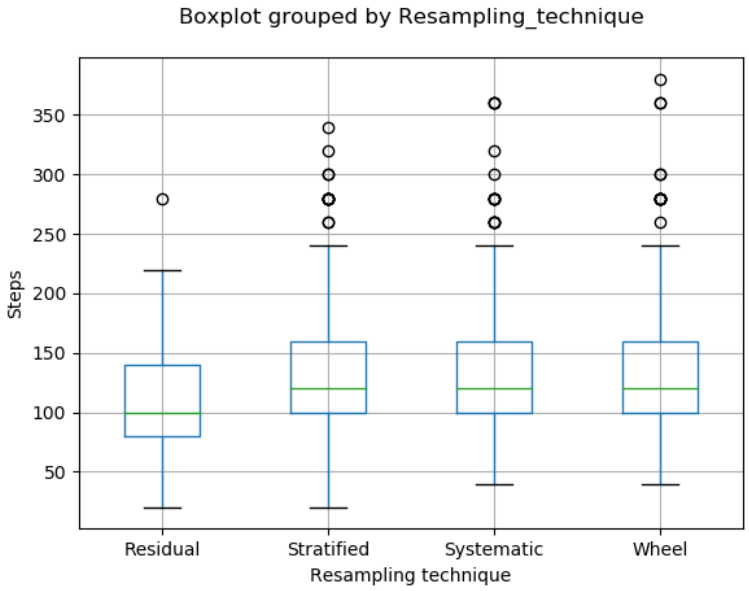


Figure 3: Number of steps before localization grouped by resampling technique.

algorithm.

5 Conclusions

After the comparison, there is no clear winner. Residual resampling is worse than the other three, but those are quite close to each other in all the metrics considered. The systematic is possibly the best by a small difference, but further analysis is required to determine it. In fact, the three techniques show quite similar results. They are so close, that it seems like the reason the success rates are not higher has nothing to do with resampling, but with the other components of the robot algorithm. So when tackling this problem, if one of these three resampling techniques are used, it is most likely that resampling is not an issue.

All of this is a reminder that, when dealing with real problems, there are a lot of things that play a significant role in the results, and not all of them can be controlled. Because of this, conclusions derived from this kind of analyses should always be carefully considered.

Also, it is quite noticeable the impact on the results that small adjustments in some parameters like *alpha* can make. This suggests that, given enough time and computational capacity to fine-tune all the parameters present in this problem, the overall improvement might be huge.

A future development regarding these analyses might be the successful implementation of a sophisticated particle number adaption technique. This feature is quite promising, because it might be able to reduce on average the number of particles and therefore the computation time, without sacrificing success rate.

A promising development, aligned with the trends in the industry, would be transforming this algorithm into a SLAM algorithm and replicating the comparative analysis. This analysis may show completely different results and conclusions, just because of the characteristics of the problem.

References

[1] I. Bukhori and Z. H. 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, no. 4, p. 17298814-1771746, 2017.

[2] I. Bukhori, Z. H. Ismail, and T. Namerikawa, "Detection strategy for kidnapped robot problem in landmark-based map monte carlo localization," in 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS), pp. 75–80, IEEE, 18/10/2015 - 20/10/2015

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] M. Lin, C. Yang, D. Li, and G. Zhou, "Intelligent filter-based slam for mobile robots with improved localization performance," IEEE Access, vol. 7, pp. 113284–113297, 2019.

[5] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics. Intelligent robotics and autonomous agents, Cambridge Mass.: MIT Press, 2005.

[6] D. Fox, "Adapting the sample size in particle filters through kld-sampling," The International Journal of Robotics Research, vol. 22, no. 12, pp. 985–1003, 2003.

[7] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), pp. 1225–1232, IEEE, 24-28 April 2000.

[8] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," IEEE Signal Processing Magazine, vol. 32, no. 3, pp. 70–86, 2015.

[9] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in 2006 IEEE Nonlinear Statistical Signal Processing Workshop, pp. 79–82, IEEE, 13/09/2006 - 15/09/2006.

[10] W. Adiprawita, A. Suwandi Ahmad, J. Sembiring, and B. R. Trilaksono, "A novel resampling method for particle filter for mobile robot localization," International Journal on Electrical Engineering and Informatics, vol. 3, no. 2, pp. 165–177, 2011.

[11] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," IEEE Transactions on Signal Processing, vol. 53, no. 7, pp. 2442– 2450, 2005.

[12] B. Balasingam, M. Bolic, P. M. Djuric, and J. Miguez, "Efficient distributed resampling for particle filters," in Efficient distributed resampling for particle filters, pp. 3772–3775, IEEE, 22/05/2011 - 27/05/2011.

[13] K. Sasirekha, Prassanna Baby, "Agglomerative Hierarchical Clustering Algorithm- A Review," 2013

[14] S. I. Martínez, J. A. C. Rocha, J. L. Menchaca, M. G. T. Berrones, J. G. Obando, J. P. Cobos, and E. C. Rocha, "An autonomous navigation methodology for a pioneer 3dx robot," Computer Technology and Application, vol. 5, no. 2, 2014.

Índice general

Índice de figuras

Índice de tablas

List of Algorithms

1 Introducción

1.1. Estructura de la memoria

Esta memoria consta de 6 capítulos: un primer capítulo de introducción , donde se explica el funcionamiento general del filtro de partículas y se mencionan algunas aplicaciones (Capítulo 1); un capítulo en que se expone el estado de la cuestión (Capítulo 2); un capítulo donde se detalla cual es el problema que se plantea y la metodología (Capítulo 3); un capítulo en el cual se aborda la selección de parámetros llevada a cabo para la experimentación (Capítulo 4); un capítulo donde se plantean los resultados del análisis comparativo (Capítulo 5) y un último capítulo donde se exponen las conclusiones y se proponen futuros desarrollos (Capítulo 6). Adicionalmente, esta memoria tiene 2 apéndices: un primer apéndice en que se explica el proceso empleado para reducir el tiempo de computación (Apéndice A) y otro apéndice en el cual se analiza la alineación de este proyecto con los *Objetivos de Desarrollo Sostenible* de la ONU (Apéndice B).

1.2. Definición y aplicaciones generales del filtro de partículas

El objetivo de este proyecto es investigar sobre los métodos de localización de robots móviles, específicamente aquellos basados en el filtro de partículas o método de Monte Carlo secuencial.

Un filtro de partículas es un algoritmo matemático recursivo no paramétrico que emplea un conjunto de muestras denominadas partículas para representar la distribución de un proceso estocástico. Este algoritmo proporciona una representación continua del estado del sistema considerando varias hipótesis al mismo tiempo. Es, por tanto, un estimador de estado con una función multimodal.

El modelo de espacio de estado del proceso puede ser no lineal en una, varias o todas sus componentes y las distribuciones de los ruidos pueden ser cualesquiera (no necesariamente gaussianas). Este algoritmo es por ello muy versátil y puede emplearse en multitud de campos como el procesado de señales, las redes de comunicación, el diagnóstico de errores en sistemas dinámicos, tanto eléctricos como mecánicos, etc. Incluso dentro de la robótica existen múltiples

problemas distintos para los que se utiliza el filtro de partículas, de manera que existen diferentes modificaciones sobre el filtro de partículas más general que ofrecen soluciones mejores en función de las características concretas del problema a resolver. Esto permite utilizar robots móviles para una variedad de aplicaciones en la industria como transporte de piezas en fábricas, gestión de almacenes o labores de mantenimiento y monitorización.

1.3. Algunos problemas dentro de la robótica

Se explican a continuación algunos de los problemas más destacados en el campo de los robots móviles.

1.3.1. Localización global

El problema de la localización global consiste en dejar el robot en un entorno conocido pero en una posición desconocida dentro de dicho entorno. Para averiguar su localización real el robot debe recopilar información de su alrededor y contrastarla con la información que previamente tenía del entorno.

1.3.2. Robot secuestrado

En este caso, el robot está en un caso similar al anterior, pero con la diferencia de que en cualquier momento durante el proceso de localización (tanto si ya se había ubicado dentro del entorno como si no) el robot puede ser desplazado a otra localización sin ningún tipo de aviso ni sin ser consciente de ello. Este problema es, por tanto, más complejo que el anterior y requiere que una vez el robot se ha localizado, esté comprobando constantemente si ha sido movido. Para esta comprobación puede ser necesario el uso de técnicas específicas de mayor o menor nivel de sofisticación en función de la aplicación [1] [2].

1.3.3. Localización y mapeo del entorno simultáneos

Más conocido por su nombre en inglés *SLAM* (Simultaneous Localization and Mapping), consiste en dejar al robot en un entorno que le es desconocido y dentro del cual tampoco conoce su posición relativa. Para resolver este problema, el robot debe al mismo tiempo moverse por el entorno siendo consciente de los movimientos que está realizando y tomando medidas del entorno (distancias a obstáculos principalmente) para generar un mapa del mismo y ser capaz de ubicarse utilizándolo [3].

1.3.4. Localización en entornos dinámicos

En muchas circunstancias, el robot se encuentra en un entorno cuya estructura general le es conocida pero en la que hay elementos que pueden cambiar su posición. Ejemplos de esto pueden ser entornos domésticos, en los que la distribución de las habitaciones no varía pero sí la posición de los muebles o en los que puede haber personas moviéndose; o entornos industriales, en los que adicionalmente respecto al caso doméstico, es habitual la presencia de otros robots en el mismo entorno realizando tareas similares o complementarias. La estrategia para la localización en este tipo de circunstancias es similar a la del caso anterior, puesto que el robot debe estar constantemente analizando el entorno en busca de cambios respecto a la idea inicial que tenía.

1.4. Comparación con otros estimadores

Además del filtro de partículas existen otros estimadores también muy utilizados en diversas aplicaciones: el filtro de histograma y el filtro de Kalman.

El filtro de histograma es un tipo de filtro de Bayes discreto, pero que a diferencia del convencional, es capaz de representar espacios de estado continuos [4]. Para ello, descompone el espacio de estado en un número finito de regiones y posteriormente asigna a cada una de ellas una probabilidad, consiguiendo así una representación del estado del sistema consistente en una función de densidad de probabilidad arbitraria.

Por otro lado, el filtro de Kalman es un tipo de filtro Gaussiano que representa el estado estimado del sistema mediante una distribución normal multivariable con dos conjuntos de parámetros: las medias μ y las covarianzas Σ [4]. Las medias representan la mejor estimación disponible del estado del sistema, mientras que las covarianzas representan la incertidumbre asociada a esta estimación. En su forma convencional, este filtro está restringido a problemas con dinámicas y medidas lineales. Sin embargo, existe una variante, el filtro extendido de Kalman, que puede resolver problemas no lineales.

Estos estimadores son en general peores que el filtro de partículas para los problemas asociados a la localización de robots móviles por varios motivos. En el caso del filtro de histograma se debe a su incapacidad de lidiar de forma natural con espacios de estado continuos (como es el caso del espacio tridimensional y las orientaciones en los robots móviles). En el caso del filtro de Kalman, incluida su versión extendida, el problema es que las funciones Gaussianas son unimodales, por lo que este estimador no puede considerar múltiples hipótesis simultáneamente (algo también necesario en los problemas de robots móviles). Sin embargo, este último tipo de estimador es óptimo bajo algunas suposiciones y más eficiente computacionalmente que el filtro de partículas, lo que dará lugar a un tipo de filtro que combina ambos algoritmos: el filtro de partículas marginalizado [5].

1.5. Funcionamiento del filtro de partículas

La versión más general del algoritmo del filtro de partículas consiste, esencialmente, en 4 pasos:

1. Inicialización
2. Asignación de pesos y remuestreo
3. Actualización del estado de las partículas
4. Finalización[1]

En el primer paso (1) se seleccionan N partículas de una distribución uniforme en el espacio de las variables consideradas. A continuación, (2) se asignan pesos a las partículas iguales a su probabilidad condicionada de representar con exactitud el estado del sistema y después se remuestrea un cierto número de partículas (típicamente N, pero no necesariamente) con reemplazamiento de forma que la probabilidad de escoger cada partícula sea de alguna forma proporcional al peso asignado. Posteriormente, (3) se actualiza el estado de las partículas

[1] Este último paso no forma estrictamente parte del algoritmo según gran parte de la literatura. Además, es dependiente del problema a tratar y los criterios varían según la aplicación, pero dada su relevancia en la localización de robots móviles se ha decidido incluirlo.

según las dinámicas estimadas del sistema en este período. Los pasos (2) y (3) se repiten alternadamente hasta que se cumple el criterio de finalización (4).
Póngase por ejemplo, un caso genérico de un robot que se mueve en el plano que intenta localizarse en un cierto entorno conocido de antemano (el robot dispone de un mapa del lugar). En el primer paso (1) el robot 'colocaría' un cierto número de partículas en el mapa con una posición y orientación muestreadas de una distribución uniforme (dentro de las posibles del robot en ese mismo mapa). Este número de partículas ha de ser suficientemente grande para 'cubrir todo el mapa', es decir, que sea muy poco probable que no se inicialice una partícula suficientemente cercana al robot (no solo en las dimensiones de posición sino también en las de orientación) y es típicamente del orden de miles de partículas para robots que se mueven en el plano (dos dimensiones de posición y una de orientación).
Posteriormente, (2) el robot tomaría una serie de medidas del entorno con distintos sensores (LIDAR, ultrasonidos, cámaras, etc) y contrastando la información de estas medidas con aquello que esperaría medir si estuviera en la posición de cada una de las partículas y decidiendo de esta manera que partículas es más probable que representen el estado del robot (esto dependerá de la fiabilidad de las medidas, es decir, el error que se les supone), asignando así los pesos en función de esta probabilidad. Después, muestrearía (con reemplazamiento) sobre el conjunto de partículas de forma que las partículas sean seleccionadas proporcionalmente a su peso, eliminando así aquellas partículas menos probables.
En el siguiente paso, (3) el robot actualizaría la posición de las partículas. Para ello, requiere de medidas que le permitan estimar cuál ha sido su desplazamiento y su cambio de orientación durante la iteración del algoritmo (estimación igualmente sujeta a un cierto error). Basándose en esta estimación, aplicará una transformación sobre las partículas desplazándolas y girándolas en la misma medida (añadiendo una componente aleatoria para representar el error asociado a la medida del desplazamiento y giro del propio robot). Las medidas pueden estar proporcionadas por sensores integrados en el robot (acelerómetros, giroscopios, velocímetros, entre otros) o por sistemas de posicionamiento externos (GPS, balizas). De esta forma, en la siguiente iteración se volverían a tomar las medidas del entorno y se repetiría el proceso.
Finalmente, (4) si el algoritmo ha funcionado correctamente las partículas tenderían tras suficientes iteraciones a concentrarse cerca de la posición del robot (siempre existe una cierta dispersión) y cuando se cumpliese un cierto criterio de compactibilidad de las partículas se daría por conocida la posición real del robot. Un ejemplo de dicho criterio sería que si se hiciera una clusterización jerárquica de las partículas el clúster más grande tuviese un porcentaje mínimo de las partículas.

1.6. Integración del filtro de partículas en un robot móvil

El filtro de partículas es un elemento crucial en los algoritmos de robots móviles. Sin embargo, este debe integrarse con otra serie de componentes para que el robot pueda operar adecuadamente. En la Figura 1.1 se representa cómo es el ciclo de comportamiento y qué elementos debe tener. El robot debe ser capaz de tomar medidas del entorno a través de sensores, procesarlas para obtener información, utilizar esta información para averiguar dónde está y qué tiene a su alrededor, y después decidir qué es lo que debe hacer (hacia donde debe dirigirse, si debe realizar alguna tarea específica, etc), y por último, tras decidirlo debe actuar, ya sea desplazándose en alguna dirección o realizando otra tarea como puede ser cargar/descargar

paquetes, emitir algún sonido de alerta, entre otras. En este ciclo, el filtro de partículas se integra en la componente que transforma las medidas de los sensores en información. Dicha información se utiliza para conocer la localización del robot, contrastándola con información que el robot ya pueda tener (mapas totales o parciales del entorno). Otro elemento importante es la navegación, que se encarga de tomar decisiones inmediatas cuando se dispone de datos de los sensores pero no de información precisa acerca de la posición. Esto es imprescindible puesto que para determinar la ubicación del robot, normalmente este necesita desplazarse y recabar información, por lo que la navegación actúa en este caso de manera independiente al filtro de partículas.

Aunque en este proyecto sólo se trata el problema de la localización, esta está relacionada con los pasos siguientes. Esto no solo es por la razón más obvia, que es que el robot no puede saber hacia donde tiene que ir si no sabe donde está, sino que también es importante tener en cuenta la precisión y la confianza en la localización para las tareas que el robot vaya a realizar después.

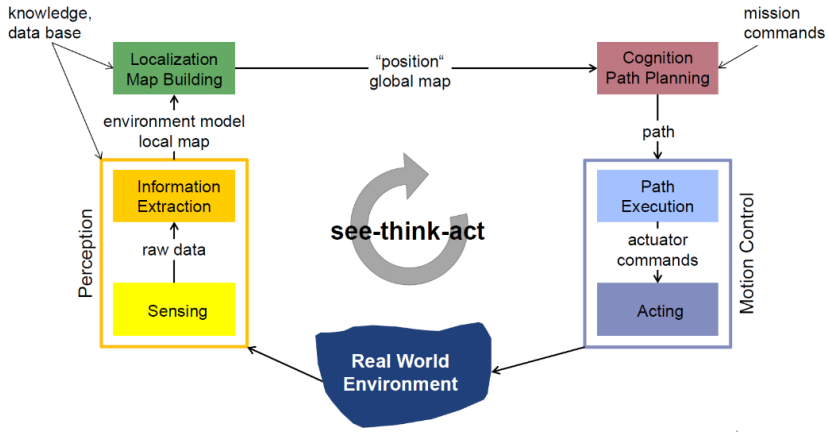


Figura 1.1. Representación conceptual del ciclo de comportamiento de un robot móvil.Fuente: Autonomous mobile robots course, Autonomous Systems Lab, ETH Zürich, Spring 2019 (https://asl.ethz.ch/education/lectures/autonomous_mobile_robots/spring-2019.html).

2

Estado del arte

Para evaluar el estado de la cuestión se van a explicar algunas de las variaciones sobre el filtro de partículas básico.

2.1. Adaptación del número de partículas

El filtro de partículas básico mantiene un número constante de partículas a lo largo de todo el proceso. No obstante, la carga computacional crece linealmente con el número de partículas, por lo que es conveniente reducir este número cuando sea posible. Varias modificaciones del algoritmo básico consisten en modificar dinámicamente el número de partículas para tratar de mantener el número de partículas necesario garantizando una cierta fiabilidad.

Una variante modifica el número en función de la probabilidad de las observaciones existentes (las partículas). Este algoritmo funciona de la siguiente manera: se prefija un umbral de probabilidad y durante el paso de remuestreo en vez de tomar la misma cantidad de partículas que había inicialmente se toman partículas y se calculan sus probabilidades una a una, de forma que se toman partículas hasta que la suma de las probabilidades excede el umbral prefijado [6]. Nótese que este algoritmo no disminuye necesariamente el número de partículas a cada paso y es determinante seleccionar el umbral correctamente para que esta modificación suponga una mejora real respecto al filtro convencional.

Otra alternativa es el muestreo KLD. Esta técnica de muestreo se basa en acotar el error de estimación introducido al representar la distribución del filtro de partículas en base a muestras. Para calcular el error de estimación se calcula la distancia Kullbach-Leibler (Ecuación 2.1) (que da nombre a la técnica, pues KLD significa *Kullbach-Leibler distance*) entre la representación de la distribución de densidad basada en muestras y la creencia de la distribución real de la función de densidad[1] [6].

$$K(p,q) = \sum_x p(x) log \frac{p(x)}{q(x)} \tag{2.1}$$

[1]Esta distancia es una técnica generalmente aceptada para medir la diferencia entre dos distribuciones, aunque en rigor no cumple las características necesarias para ser una métrica. La distancia KL es no-negativa y solo es cero si ambas distribuciones son idénticas, pero no es simétrica y no cumple la propiedad triangular.

Esta técnica permite garantizar con una cierta probabilidad que el error entre la función de densidad real y la función de densidad basada en muestras es menor que una cierta cota prefijada. Rigurosamente, permite determinar el número de muestras necesario para que con una cierta probabilidad $1 - \delta$, el error de aproximación entre ambas distribuciones sea menor que ϵ, donde δ y ϵ son valores prefijados. Para ello, toma muestras de la función de densidad (es decir, añade partículas) hasta que la diferencia entre la distribución basada en muestras y la real aproximada es menor que la cota mencionada anteriormente [6].

2.2. Filtro de partículas marginalizado

Uno de los problemas del filtro de partículas convencional es que su coste computacional crece exponencialmente con el número de dimensiones en el espacio de estado, puesto que si se añade una nueva dimensión las partículas deben cubrir con la misma densidad las dimensiones existentes y además la nueva, lo que requeriría generar partículas proporcionalmente a N en el nuevo caso por cada partícula existente antes, de forma que N sea suficiente para cubrir con ciertas garantías el espacio de la nueva dimensión introducida.

Este problema se puede resolver bajo ciertas circunstancias con el filtro de partículas marginalizado o Rao-Blackwellizado[2]. Como ya se ha explicado la potencia del filtro de partículas reside en su capacidad de resolver problemas no-lineales y no-gaussianos, mientras que el filtro de Kalman puede resolver analíticamente problemas lineales gaussianos (y no-lineales gaussianos en su versión extendida). Cabe destacar que esta resolución analítica es menos costosa computacionalmente que la del filtro de partículas en los casos en que puede aplicarse. Así, en aquellos casos en los que dentro del espacio de estado existe un subespacio susceptible de estimarse con un filtro de Kalman, es posible dividir el espacio de estado en dos subespacios, el lineal y el no-lineal. De esta forma, es posible resolver la componente lineal con un filtro de Kalman y la componente no-lineal con un filtro de partículas convencional, consiguiendo así reducir coste computacional global. Empleando este método se consigue también mejorar la estimación, pues para el caso lineal el filtro de Kalman es un estimador óptimo, por lo que la estimación realizada con esta variante será mejor (tendrá una menor varianza). Un caso de uso de esta variante es el posicionamiento asistido por información del terreno de aeroplanos Dicho problema es altamente no-lineal porque utiliza una base de datos de elevaciones, lo que impide el uso del filtro extendido de Kalman; y es un problema de dimensionalidad muy elevada (9 dimensiones), lo que impide el uso del filtro de partículas. Sin embargo, es ideal para el filtro de partículas marginalizado, puesto que solamente 2 dimensiones son no-lineales [5] [7].

2.3. Introducción de nuevas partículas

Uno de los problemas a resolver por los filtros de partículas es el conocido como 'secuestro', en el que el robot puede ser movido de posición en cualquier momento. Si dicho secuestro se produce cuando el filtro de partículas ya ha reducido las posibilidades de la ubicación del robot a un número pequeño de posibles localizaciones dentro del mapa el filtro de partículas básico no puede resolver dicho problema salvo que eventualmente una de las creencias del filtro original coincida con el estado real del robot, lo que no siempre tiene por qué ocurrir. Para resolver este problema la fórmula que se plantea es sencilla: a cada iteración se añade un cierto número de partículas independientes de las que ya existen (es decir, se inicializan nuevas

[2]Llamado así porque incorpora el teorema de Rao-Blackwell.

partículas y se *añaden al mapa*). De esta manera, si en algún momento el robot es secuestrado el filtro puede recuperarse en un número relativamente bajo de iteraciones, dependiendo del número de partículas añadidas. Además, incluso aunque el robot no se enfrente específicamente a este problema, la inclusión constante de nuevas partículas proporciona una mayor robustez al algoritmo, aunque conlleva un coste añadido en carga computacional tanto por el mayor número de partículas como por la constante inicialización de nuevas partículas [8].

2.4. Remuestreo

El objetivo del remuestreo es evitar o minimizar la degeneración de las partículas durante el proceso. Dicha degeneración consiste en la concentración de la mayor parte del peso en un número muy reducido de partículas en unas pocas iteraciones, teniendo por consiguiente un elevado número de partículas con un peso despreciable. Existen multitud de técnicas de remuestreo con diferentes características. Se explican a continuación varias de ellas, divididas en dos grupos: técnicas tradicionales y variaciones sobre las técnicas tradicionales, todas ellas recogidas en [9] excepto el remuestreo de la rueda. Otras fuentes que pueden resultar de interés al lector en esta cuestión son [10] [11].

2.4.1. Técnicas tradicionales

2.4.1.1. Multinomial

Es quizá la técnica más intuitiva. Se generan N número aleatorios independientes en una distribución uniforme $(0, 1]$. Después se utiliza cada uno de ellos para seleccionar una partícula proporcionalmente a su peso. Este método es muy sencillo pero con una varianza muy elevada (que favorece la degeneración de partículas) y extremadamente ineficiente computacionalmente, por ello en la práctica no se utiliza.

2.4.1.2. Estratificado

Se divide el conjunto de partículas en subconjuntos (estratos), dividiendo el intervalo $(0, 1]$ en N intervalos disjuntos y seleccionando un número aleatorio en cada uno de ellos. Después, se seleccionan las partículas de la misma manera que el caso multinomial. Este método tiene menor varianza que en el caso multinomial y es más eficiente computacionalmente, aunque sigue teniendo la necesidad de generar N números aleatorios, que es costoso. Además, este método garantiza que se seleccionan todas las partículas de peso normalizado mayor que $2/N$, pues estas partículas ocuparán al menos un intervalo entero.

2.4.1.3. Sistemático

Se trata de una variante del remuestreo estratificado en el que solo se genera un número aleatorio entre $u_1 = (0, 1/N]$ y los demás números se calculan como $u_n = u_1 + \frac{n-1}{N}$. Esta variante requiere de un único número aleatorio, por lo que es ligeramente menos costosa computacionalmente.

2.4.1.4. Residual

Se generan réplicas de todas las partículas de peso normalizado mayor que $1/N$, siendo $n_i = \lfloor Nw_i \rfloor$ el número de réplicas de la partícula i-ésima, donde w_i es el peso normalizado de la partícula correspondiente. Después, se remuestrean $m = N - \sum n_i$ partículas del conjunto

de partículas inicial utilizando los residuos de los pesos en vez de los pesos originales. Dichos residuos se calculan como $\hat{w}_i = w_i - \frac{n_i}{N}$. Para esta selección se puede emplear uno de los otros métodos de remuestreo, típicamente el multinomial.

2.4.1.5. De la rueda

Existen múltiples variantes del remuestreo de la rueda. En este apartado se incluye el pseudocódigo de la que se emplea en este proyecto y a la que se hará referencia como remuestreo de la rueda (Algoritmo 2.1).

Algorithm 2.1. Remuestreo de la rueda

```
index = U(1, N)
beta = 0
for i = 1 : N do
    beta = beta + U(0, 2 * max(weights))
    while weights[index] < beta do
        beta = beta - weights[index]
        index = index + 1
    end
    draw(particle[index])
end
```

Esta variante es sabido que reduce la varianza respecto del remuestreo multinomial, aunque no existe una justificación analítica para ello. Intuitivamente, el hecho de que en cada vuelta de la rueda se recorran todas partículas impide que una partícula o un número reducido de ellas se repliquen un número elevado de veces por elevados que sean sus pesos. Además, el hecho de que en cada iteración del bucle *For* se añada a beta el doble del máximo de los pesos permite *saltarse* los pesos más elevados en cada vuelta de la rueda, evitando que se repliquen demasiadas veces. Así pues, este algoritmo favorece la réplica de aquellas partículas de menor peso, lo cual reduce la varianza y la degeneración de partículas.

2.4.2. Variantes sobre las técnicas tradicionales

2.4.2.1. Remuestreo residual sistemático

Este método combina los métodos de remuestreo residual y sistemático. Generando un número aleatorio en la uniforme $(0, 1/N]$ y utilizándolo para evitar el segundo bucle del método residual, de forma que el primer peso se evalúa en comparación con el número generado y este se va actualizando en cada iteración.

2.4.2.2. Remuestreo compuesto

Existen también otras técnicas en las que las partículas se dividen en grupos con criterios predefinidos y se tratan de forma distinta en función de en qué grupo estén. Algunos de estos métodos emplean umbrales de peso según los cuáles agrupan las partículas, que pueden ser a su vez estáticos o dinámicos. Otros consideran otro elemento aparte del peso de las partículas, la información de su estado. De esta manera, partículas que están muy juntas pueden representar un mismo estado, de forma que se fusionan para reducir el número de partículas distintas. De manera contraria, algunos métodos fragmentan aquellas partículas suficientemente grandes en varias partículas más pequeñas distribuidas alrededor de esta (manteniendo en todos los casos la suma de los pesos).

2.4.2.3. Remuestreo modificado

Otra técnica consiste en remuestrear partículas recalculando sus pesos como funciones de los pesos originales. De esta forma es posible *beneficiar* a aquellas partículas más grandes o más pequeñas según se desee aumentar la diversidad o acelerar la convergencia.

En el filtro de partículas convencional el peso de cada partícula es directamente proporcional a su probabilidad, siendo igual a esta en caso de no normalizar los pesos. Sin embargo, una variante habitual consiste en calcular el peso de cada partícula como función de su probabilidad según la ecuación no lineal Ecuación 2.2. Esta variante permite favorecer a las partículas de menor peso si se elige α menor que 1 o a las de mayor peso si se elige mayor que 1 (en la versión convencional este parámetro es igual a 1, pues el exponente se omite).

$$\omega_i = (p_i)^{\alpha} \tag{2.2}$$

2.4.2.4. Remuestreo distribuido

Otra característica que se ha de considerar para evaluar la utilidad de un algoritmo no es solo la complejidad computacional, sino también su capacidad de ejecutarse en paralelo, que en muchas ocasiones puede resultar más beneficioso y económico que disminuir la complejidad. En esta línea existen también algunas variantes en las que se intenta evitar la normalización de los pesos, de forma que tras un paso inicial de asignación de las partículas, sea posible ejecutar el proceso en paralelo. Dentro de este tipo de algoritmos se distinguen dos tipos: aquellos que distribuyen las partículas de forma proporcional y de forma no proporcional [12] [13].

3

Planteamiento del problema

A continuación se describe cuál es el problema específico que se va a abordar en el análisis comparativo: que algoritmos se van a comparar, qué metodología se va a emplear y en qué entorno y con qué medios se va a realizar el análisis.

3.1. Objetivos

El propósito de este proyecto es estudiar varias variantes de algoritmos de remuestreo en filtros de partículas y determinar si existen unas variantes mejores que otras, cuál es la mejor variante para el problema de la localización global que se plantea y cuantificar cuál es el efecto resultante de utilizar unas variantes y no otras. Las variantes de remuestreo que se van a comparar son estratificado, sistemático, residual y de la rueda (todas explicadas en la Sección 2.4).

3.2. Metodología

Para la comparación se evalúan los resultados de cada una de las técnicas de remuestreo en un problema de localización global en dos dimensiones en un simulador, en el se sitúa un robot en un entorno conocido pero en una posición desconocida y este debe moverse por dicho entorno y recabar información hasta determinar cuál es su posición. Cada *viaje* del robot desde un punto inicial hasta que se localiza se denominará *expedición* en este proyecto.

Se evalúan el índice de éxito en la localización y la precisión de localización en las expediciones exitosas, así como otras variables que puedan resultar de interés para evaluar las diferencias entre los algoritmos de remuestreo. Una expedición se define como exitosa si el error de localización en distancia es menor de 65 centímetros y el error de localización en ángulo es menor de 0.7 radianes. Además, una expedición se da por inválida (no se tiene en cuenta) si la duración es superior a 500 iteraciones del algoritmo del robot.

Para afrontar el problema de localización global es necesario no sólo un algoritmo de remuestreo sino un algoritmo de navegación para el robot, un filtro de partículas completo y un criterio de finalización. Sin embargo, se parte de la situación en que estos elementos se

encuentran desarrollados y funcionan razonablemente bien, de forma que permiten centrar el proyecto en evaluar los resultados fruto de emplear los diferentes algoritmos de remuestreo. Las únicas modificaciones realizadas a estas componentes se centran no en la calidad de la ejecución de las expediciones, sino en la automatización de las simulaciones (esto es, permitir que se puedan realizar tantas simulaciones como se quiera, unas a continuación de otras sin necesidad de intervención humana) y en la reducción de los tiempos de ejecución, que inicialmente eran demasiado elevados e impedían realizar cantidades elevadas de simulaciones para conseguir muestras de datos suficientemente grandes.

Para intentar mostrar los algoritmos en las mejores condiciones, se analizan los efectos de cambiar varios parámetros de estas componentes y se emplean para la comparación aquellos que dan mejores resultados. No es posible analizar todos los parámetros debido a que requeriría demasiado tiempo, por lo que se elige un grupo reducido de ellos que tienen un impacto más directo en el propio remuestreo y se mantienen el resto con los valores de partida.

Para evitar sesgar el análisis comparativo utilizando uno de los 4 algoritmos a comparar en la selección, el análisis de estos parámetros se realiza con la técnica de remuestreo multinomial, cuyo análisis no tiene interés en este proyecto por lo comentado en la Sección 2.4.

Cabe destacar que los valores de partida de los parámetros que no se modifican se han determinado de forma experimental utilizado el remuestreo de la rueda, por lo que existe un pequeño sesgo en favor de esta técnica.

3.3. Software y hardware utilizados

3.3.1. Software

El algoritmo responsable de la lógica de la navegación del robot, el filtro de partículas y criterio de finalización está programado en *Python 3.7*. Este algoritmo se comunica de forma síncrona a través de una API con el simulador de robots *V-REP PRO EDU 3.6.2*. El robot que se emplea es el *Pioneer 3-DX*, que dispone de 16 sónares con un alcance de un metro (aproximadamente) y tiene dos ruedas (lo cuál resulta relevante porque le permite rotar sobre sí mismo sin desplazar su centro de masas). Este tipo de robots son bastante populares en la investigación, no solo en experimentos simulados sino también reales [14]. En este proceso de comunicación el algoritmo en *Python* actúa como el *cerebro* del robot, de modo que recibe las medidas de los sensores, las procesa y determina los mandos de los actuadores, los dos motores de las ruedas. Todo esto ejecutado en un sistema operativo *Windows 10.0*.

3.3.2. Hardware

El equipo utilizado para ejecutar las simulaciones es un ordenador portátil *Lenovo Yoga C930* con 16 GB de memoria RAM y un procesador de 8 núcleos *i7-8550U*.

3.4. Problema de localización

Con el fin de limitar el coste computacional, el problema que se ha afrontado en este trabajo es una versión simplificada del problema canónico de la localización del robot en un entorno bidimensional. En esta versión del problema, todas las paredes del entorno del robot están formadas por segmentos horizontales o verticales (en la vista de planta del entorno de la Figura 3.1). Esto permite que la orientación inicial del robot pueda limitarse a 4 valores (0,

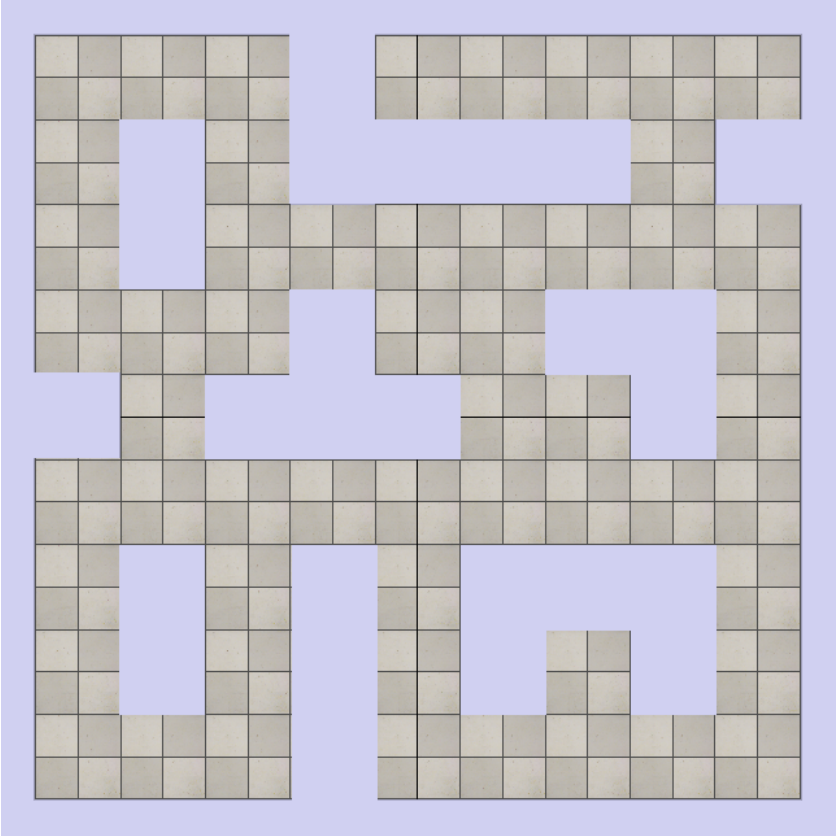


Figura 3.1. Entorno donde navega el robot.

90, 180 y 270 grados en sistema sexagesimal), puesto que el robot podría inicialmente girar sobre sí mismo (no es imprescindible que el giro sea sobre sí mismo siempre y cuando disponga de espacio de maniobra, pero sí es ventajoso) y situarse en paralelo o perpendicular a una pared visible y su orientación sería necesariamente igual a una de estas con un margen de error muy bajo. Esta simplificación del problema permite que la generación inicial de partículas solo considere dos variables continuas (posición en dos ejes) y considere la orientación como una variable discreta, limitando considerablemente el número de partículas necesarias.

A continuación se especifican los parámetros del algoritmo de localización más relevantes de los que se parte y las características del entorno donde se mueve el robot.

3.4.1. Características del entorno

El entorno donde navega el robot tiene un contorno exterior con forma cuadrada de 9 m x 9 m. El entorno se divide en 81 casillas de 1 metro cuadrado. Hay dos clases de casillas: vacías, aquellas por donde el robot puede circular, y sólidas, es decir, paredes. Se muestra en la Figura 3.1 el entorno visto en planta.

3.4.2. Parámetros

- Período de muestreo de los sensores y de comando de los actuadores: 50 milisegundos
- Período de ejecución del filtro de partículas, y por consiguiente del remuestreo y el criterio de finalización: 1 segundo

- Ruido asociado a los sensores sónar: 0,6 metros
- Ruidos asociados a la medición de v y w: 0,15 m/s y 0,2 rad/s respectivamente

Mención aparte merecen el algoritmo de navegación y el criterio de finalización, que son dos componentes con varios parámetros cada una. La primera se omite dada su extensión y complejidad y debido a que no tiene relación directa con el remuestreo, objeto de este análisis. Respecto al criterio de finalización, se emplea clusterización aglomerativa empleando como distancia entre grupos la media de las distancias y un umbral de distancia de 0,65 m (que da lugar al criterio de éxito de una expedición de la Sección 3.2), a partir del cual dos clústeres no se juntan. Por simplicidad la clusterización se realiza solo en base a la distancia euclídea de las muestras, sin considerar su orientación. Sin embargo, sería más riguroso emplear una métrica que considere también esta componente. El criterio de finalización consiste en el cumplimiento de una doble condición: el clúster de mayor tamaño (que representa la mejor estimación de la localización del robot) debe contener más del 90 % de las partículas, y el segundo de mayor tamaño debe contener menos del 5 % de las partículas.

4

Selección de parámetros

Con el fin de reducir el número de simulaciones necesarias en la comparación, se han prefijado algunos parámetros tras analizar qué valores dan mejores resultados en el remuestreo. Estos parámetros son: *alfa*[1], el número de partículas y la técnica de adaptación del número de partículas. Para evitar sesgar el análisis pero aun así poder mostrar los algoritmos en su mejor versión (es decir, con aquellos parámetros que den mejores resultados) la selección de estos parámetros se ha realizado ejecutando simulaciones empleando un algoritmo de remuestreo que posteriormente se excluye del análisis comparativo. Para ello se ha empleado el remuestreo multinomial, puesto que carece de interés para el estudio comparativo. Esta técnica de remuestreo maximiza la varianza, por lo que introduce un pequeño sesgo en favor de las técnicas con varianza más elevada, y en particular, para la selección de parámetros, introduce un sesgo en favor de alfas menores que 1 y de números de partículas relativamente elevados. En el análisis referente a los parámetros a continuación se muestran algunos gráficos en los que, aunque se aprecian las tendencias correspondientes a la teoría, estas no siempre son monótonas debido a que el número de experimentos realizado es relativamente reducido.

4.1. Selección del valor de alfa

Como se ha explicado anteriormente, el parámetro *alfa* toma un valor por defecto de 1 en el algoritmo convencional. Por ello, para seleccionar el valor se han ejecutado simulaciones en idénticas condiciones variando únicamente este valor entre 0,7 y 1,3 en saltos de 0,1. Se ha decidido mantener valores cercanos a 1 porque el valor de alfa implica un compromiso entre velocidad de convergencia e índice de éxito de la localización. El número de expediciones que se realizan con cada uno de los valores de alfa es de 25 con un número de partículas constante de 2000, que puede parecer un número reducido pero el tiempo de simulación que se ha requerido es de aproximadamente 12 horas.

El índice de éxito, observado en Figura 4.1 muestra una tendencia ligeramente decreciente en función de alfa. Aun considerando lo reducido de la muestra, se aprecia claramente que los valores menores de alfa de 0,7 y 0,8 logran unos índices de éxito notablemente superiores al

[1]Dado que *alfa* es un parámetro que se va a modificar, los algoritmos a comparar no serían las versiones básicas de los mismos, sino versiones modificadas (Sección 2.4.2.3).

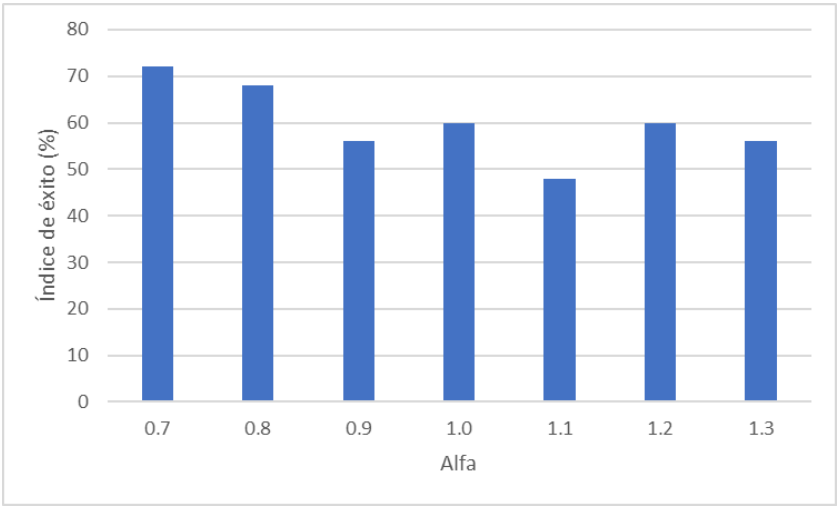


Figura 4.1. Índice de éxito en la localización en función del valor de alfa.

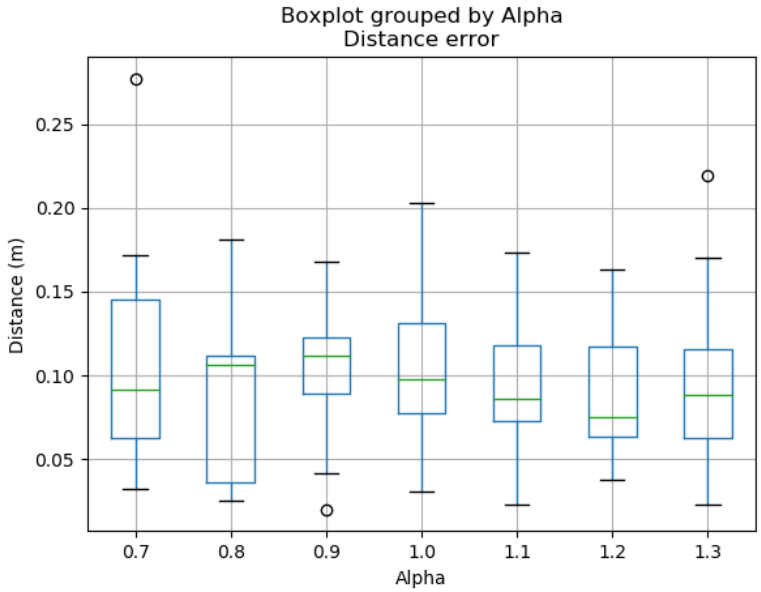


Figura 4.2. Distancia entre la localización real y la estimada en función del valor de alfa.

resto, de casi 10 puntos porcentuales más. Esto coincide con el propósito de alfa mencionado en el estado del arte (Sección 2.4.2.3) de combatir la degeneración de partículas, que permite conseguir mejores resultados.

Observando ahora el error de localización (que considera solo aquellos casos en los que la localización es exitosa) mostrado en la Figura 4.2, se comprueba que el efecto de alfa en esta variable es mínimo, siendo los valores en todos los casos aceptables para aplicaciones de propósito general (hablamos de un error menor que 0,2 m en casi todos los casos en un entorno con una superficie de más de 80 metros cuadrados).

En lo referente al número de iteraciones hasta la localización, se observa en la Figura 4.3 que este valor no guarda una relación clara con alfa. En teoría, la tendencia debería ser ligeramente decreciente puesto que los alfas elevados favorecen las partículas de peso elevado y la rápida convergencia. Hay varias razones posibles por las que esta relación no es clara en este caso: la primera es que los índices de éxito no son muy elevados en general, debido entre otras cosas

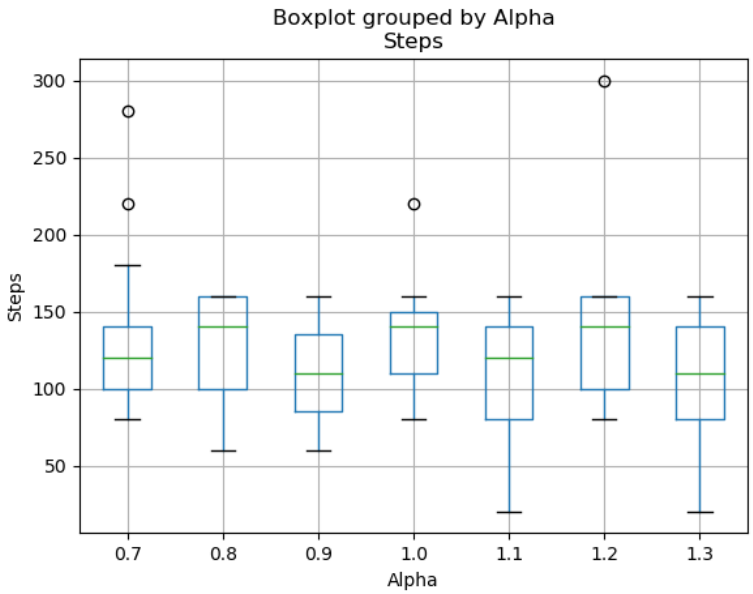


Figura 4.3. Iteraciones del algoritmo hasta la localización.

a que el muestreo multinomial es inferior al resto de técnicas mencionadas; la segunda es que existe un sesgo relacionado con lo anterior, y es que solo se consideran los casos exitosos, en los que el número de iteraciones promedio hasta la localización (acertada o no) no es necesariamente igual a los casos fallidos; la tercera es que las variaciones entre los valores de alfa pueden no ser excesivas a nivel práctico, y si se combina esto con el hecho de que el tamaño de la muestra es relativamente reducido por las necesidades de computación, la tendencia teórica no aparece reflejada en este caso.

Considerando los resultados mostrados en los gráficos, parece que los valores de alfa menores que 1 parecen claramente mejores en este caso. De entre los valores, analizados parece que el que mejores resultados aporta es el menor de ellos, 0,7, seguido de cerca del valor 0,8. Sin embargo, para combatir el sesgo existente de la menor varianza de los algoritmos a comparar frente al muestreo multinomial empleado para este análisis, y reducir el tiempo de ejecución de las iteraciones se emplea el valor de 0,8.

4.2. Selección del número inicial de partículas

Para la selección del número de partículas se han realizado simulaciones con valores entre 1500 y 4000 en saltos de 500. De manera similar al caso anterior, el número de partículas implica un compromiso entre el tiempo de ejecución y el índice de éxito. El número de expediciones realizadas con cada valor es de 60 y el tiempo de simulación necesario ha sido aproximadamente 42 horas. Se ha empleado un valor de *alfa* de 1 y el número de partículas se ha mantenido constante. Nótese que se ha dedicado más tiempo no solo porque se hayan realizado más expediciones, sino también porque al realizarse con más partículas el tiempo promedio es mucho mayor.

En la Figura 4.4 se aprecia la tendencia creciente del índice de éxito en función del número de partículas, de la misma forma que en la Figura 4.5 se observa la misma tendencia en el tiempo de ejecución de las iteraciones, dando lugar al compromiso mencionado anteriormente.

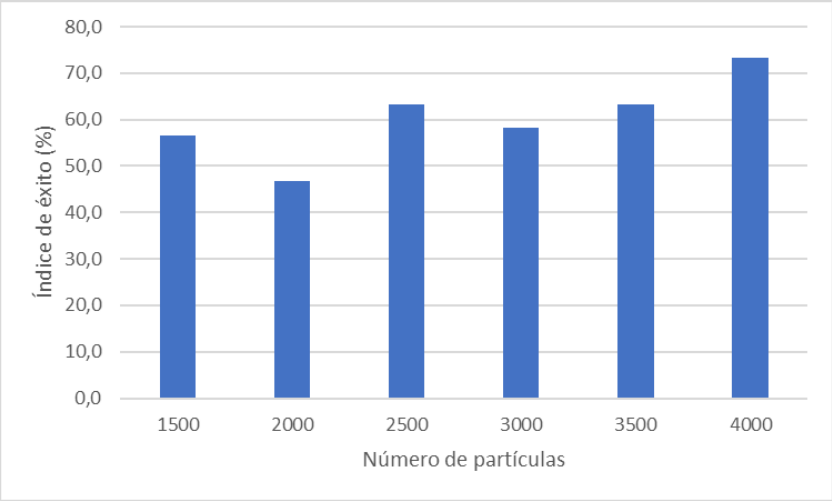


Figura 4.4. Índice de éxito en función del número de partículas.

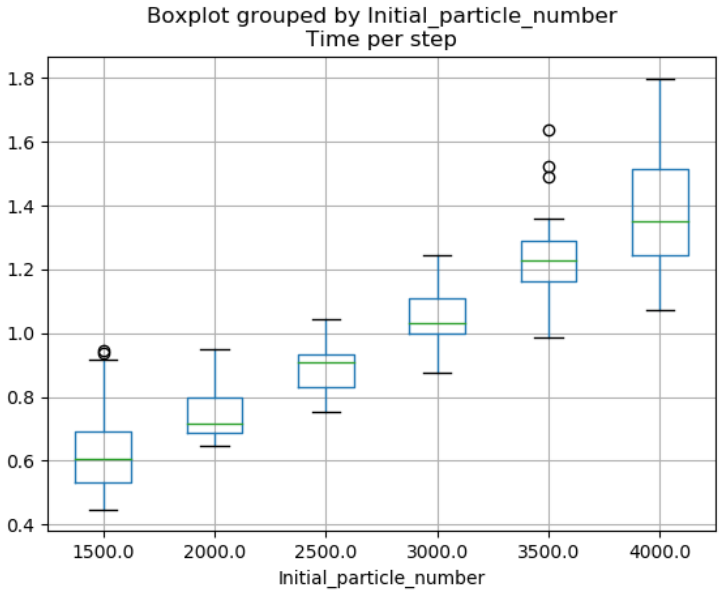


Figura 4.5. Tiempo de ejecución de cada iteración en función del número de partículas.

Se toma para el análisis comparativo el valor intermedio de 2500 para el número inicial de partículas, puesto que aunque el las aplicaciones industriales el índice de éxito es muy relevante, un menor tiempo de ejecución permitirá en este caso tener una muestra mayor para poder comparar con fiabilidad los distintos algoritmos, y aunque los índices de éxito de estos sean menores, por tratarse de una comparación, esto es menos relevante que tener una muestra suficientemente grande para que los resultados sean consistentes.

4.3. Adaptación dinámica del número de partículas

Para la adaptación del número de partículas se han evaluado 3 técnicas: no modificar el número de partículas, decrecimiento constante del número de partículas en un 5 %[2] en cada remuestreo y adaptación en función de la probabilidad de las partículas muestreadas Sección 2.1.

[2]Este valor da buenos resultados experimentalmente, aunque profundizar en el análisis del valor óptimo para esta variante es una posible vía de mejora del filtro de partículas.

Esta última opción requiere de la selección de un umbral de probabilidad y potencialmente de la selección de valores mínimos y máximos para el número de partículas.

Dicho umbral de probabilidad se ha ajustado de manera experimental para mantener valores del número de partículas de entre 2000 y 3000 la mayor parte del tiempo. Sin embargo, se ha observado que sin razón aparente el número de partículas puede variar enormemente de una iteración a otra, pasando de unos pocos miles a varios millones de partículas, un número que es inmanejable con la capacidad de computación utilizada. Para afrontar este problema se han establecido umbrales mínimo y máximo para el número de partículas, con la esperanza de que tras un número bajo de iteraciones el número de partículas volviese a los valores habituales. Sin embargo, con frecuencia el número de partículas se mantenía constante en los umbrales y aun cuando las partículas se encontraban en el umbral máximo, los resultados no reflejaban ninguna mejora visible respecto al número de partículas constante. La razón de que esta técnica haya dado resultados pobres en comparación incluso con el número de partículas constante es, aparentemente, que el número de partículas que se maneja en este experimento es relativamente bajo por lo que las variaciones estadísticas de una iteración a otra pueden suponer la saturación del número de partículas a uno de los umbrales, tras lo cual es difícil que el número de partículas vuelva a los valores *razonables* de entre 2000 y 3000. Este es posiblemente uno de los sacrificios de la experimentación en favor de la dimensionalidad reducida del experimento. Por estas razones, el uso de esta técnica se ha descartado para el resto de la experimentación.

Para comparar las dos técnicas restantes se han simulado 120 expediciones con cada una (este número es sustancialmente mayor porque se trata solo de 2 variantes) empleando un valor de alfa de 1 y un número de partículas inicial de 2000. En total el tiempo de simulación necesario ha sido de aproximadamente 16 horas.

En las imágenes Figura 4.6 y Figura 4.7 se observan dos cosas interesantes: la primera es que el índice de éxito cuando se disminuye el número de partículas en cada iteración es considerablemente superior a la alternativa (10 puntos porcentuales). Esto podría tratarse de una anomalía estadística debido a lo reducido de la muestra pero también podría reflejar que puede resultar beneficioso forzar la convergencia más rápida del algoritmo en ocasiones, para evitar que tras muchas iteraciones las variaciones estadísticas hagan que posiciones inicialmente menos probables y erróneas acaben prevaleciendo sobre la posición correcta, pues según aumentan las iteraciones menos relevancia se da a la información recogida en el inicio. La segunda, en línea con la teoría, muestra como aunque los tiempos de ejecución promedios son muy similares en ambos casos, cuando se fuerza la disminución de partículas la varianza se reduce considerablemente puesto que aunque los tiempos de ejecución mínimos son muy similares los máximos son inferiores debido a que tras muchas iteraciones el número de partículas se ha reducido de forma considerable y es mucho más probable que el algoritmo converja. Por ello, se realiza el estudio comparativo forzando una reducción del 5 % del número de partículas en cada remuestreo.

4.4. Resumen de la selección

Los parámetros seleccionados se encuentran resumidos en la Tabla 4.1.

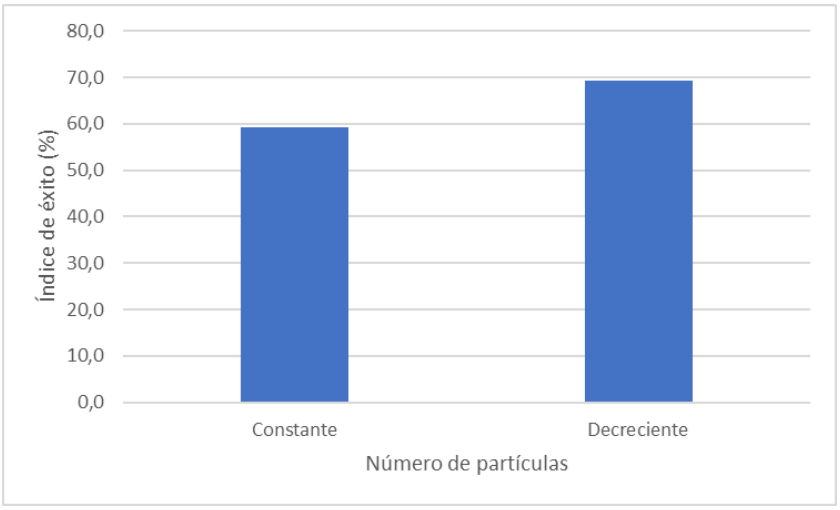


Figura 4.6. Índice de éxito en función de la adaptación del número de partículas.

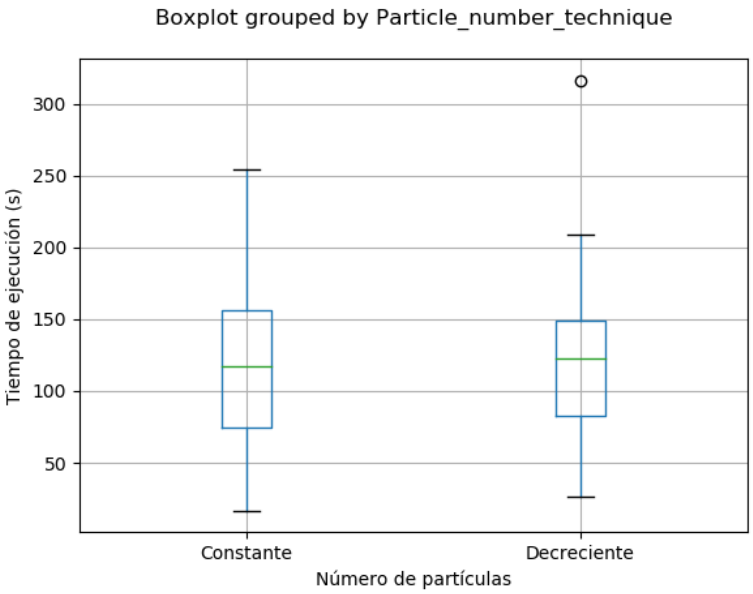


Figura 4.7. Tiempo de ejecución de cada iteración en función de la variación del número de partículas.

Tabla 4.1. Resumen de los parámetros.

Parámetro	Valor final
Alfa	0,8
Número de partículas	2500
Estrategia de adaptación	Decreciente 5 %

5

Comparación de las técnicas de remuestreo

Se comparan a continuación las 4 técnicas de remuestreo estudiadas: residual, estratificado, sistemático y de la rueda. Para generar los datos necesarios para evaluarlas se han ejecutado 500 expediciones con cada una de ellas desde 100 orígenes aleatorios distintos dentro del entorno (repitiendo la expedición desde cada origen 5 veces para que los resultados sean más consistentes). Los orígenes que se han empleado se muestran en la Figura 5.1. En la El tiempo total de ejecución dedicado ha sido de aproximadamente 36 horas por técnica de remuestreo, esto es aproximadamente 6 días en total.

En la Figura 5.2 se observa como los índices de éxito de 3 de las técnicas son muy similares (todos ellos entre 80 y 83 %), mientras que el del remuestreo residual es muy inferior (menos de 65 %), más similar al observado en el capítulo Capítulo 3, empleando el remuestreo multinomial. Parece que esos 3 algoritmos son muy fiables y que el índice de éxito tiene una cota superior de aproximadamente del 85-90 % debido, no al remuestreo, sino al resto de parámetros que determinan el comportamiento del robot (algoritmo de navegación, precisión de los sensores y actuadores, criterio de finalización, etc).

Observando la Figura 5.3[1] se ve como una vez más las técnicas de remuestreo estratificado, sistemático y de la rueda dan resultados similares (de hecho son casi idénticos, pues las medias y los límites de la caja son iguales y la longitud de los bigotes es muy similar), mientras que el remuestreo residual es ligeramente más rápido en converger. Esto es consistente con lo observado en Figura 5.2, puesto que refleja el compromiso existente entre el índice de éxito y la velocidad de convergencia. Sin embargo, para la inmensa mayoría de aplicaciones una reducción del número de iteraciones de menos del 20 % no parece suficiente mejora para sacrificar más de 15 puntos porcentuales de índice de éxito. Por ello, se ignora el remuestreo residual para el resto del análisis.

En la Figura 5.4 y la Figura 5.5 se observa como los errores de localización en distancia y ángulo son también muy similares entre ellos, siendo las variables mostradas hasta el momento

[1] Nótese que la presencia de *outliers* es mucho mayor que en los casos anteriores debido a que el tamaño de la muestra es mucho mayor y no a que los valores estén más dispersos.

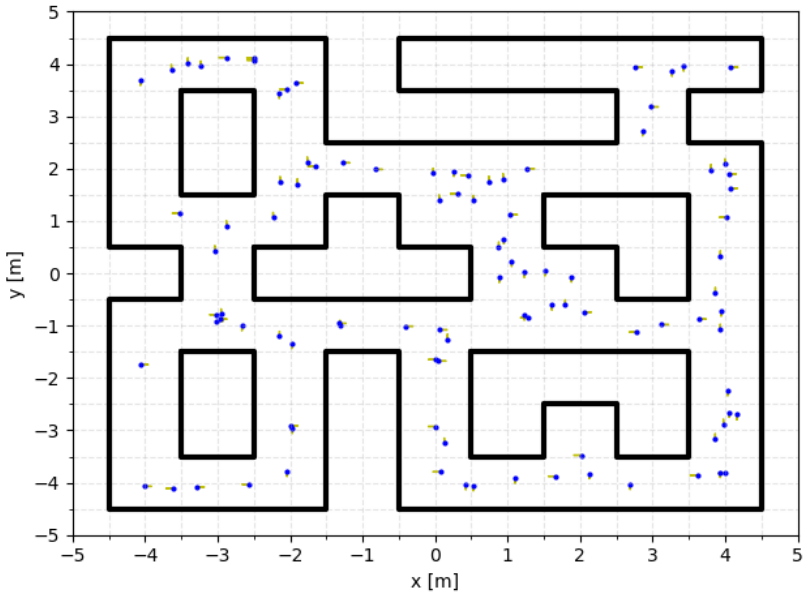


Figura 5.1. Orígenes de posición y orientación empleados en la comparación.

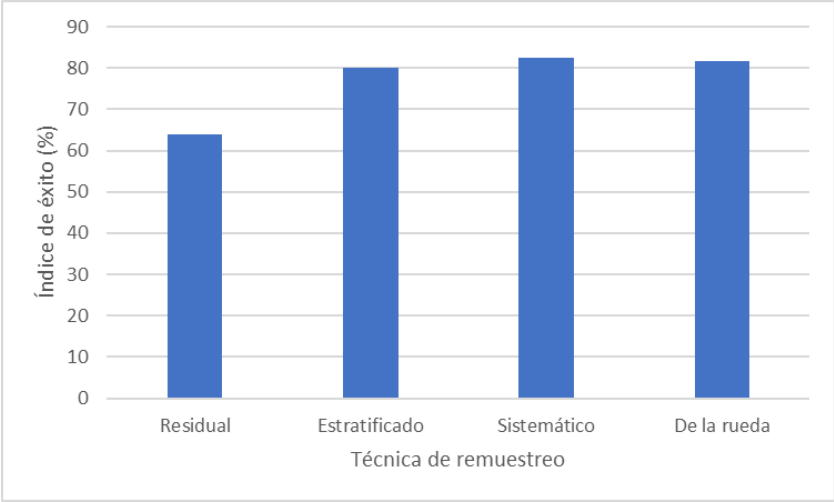


Figura 5.2. Índice de éxito de las diferentes técnicas de remuestreo.

insuficientes para afirmar que una de las técnicas de remuestreo sea claramente mejor que las demás.

Se analiza en la Figura 5.6a y la Figura 5.6b la varianza en los ejes X e Y del clúster de mayor tamaño cuando se produce una localización exitosa (el clúster que aproxima la posición y orientación del robot). En la figura se observa como las varianzas tanto en el eje X como en el eje Y son muy similares en los 3 algoritmos considerados [2].

En cambio, si se observan las varianzas representadas en la Figura 5.7a y la Figura 5.7b, se aprecia que en el remuestreo de la rueda la media de la varianza es similar pero la cola superior de la distribución contiene muchos más valores. Y esta diferencia es de hecho, relativamente grande si se observan los diagramas de cajas. Además, esta diferencia no depende del número

[2]La diferencia entre el eje X e Y es irrelevante dado que estos ejes están puestos de forma arbitraria en el entorno (e.g. no representan ninguna característica física). Esta diferencia se debe simplemente a la disposición del mapa en que se mueve el robot, pero sería prácticamente nula si el entorno estuviera generado aleatoriamente.

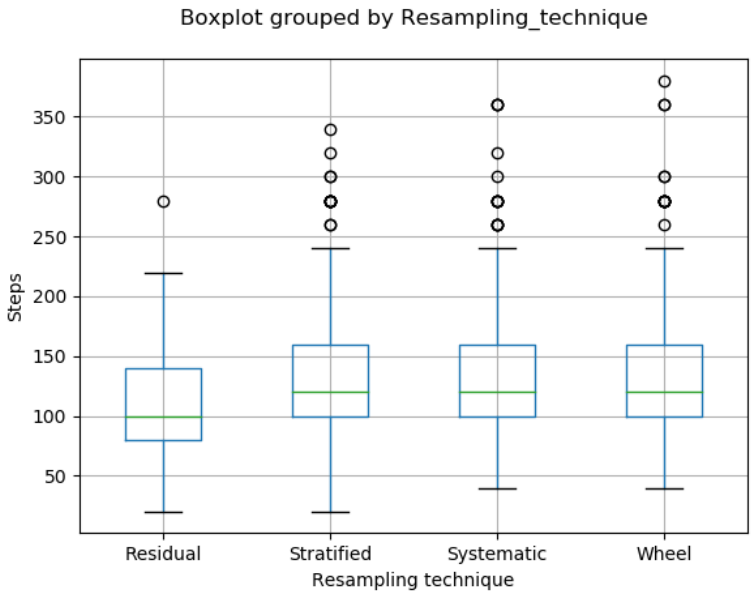


Figura 5.3. Número de iteraciones en función de la técnica de remuestreo empleada.

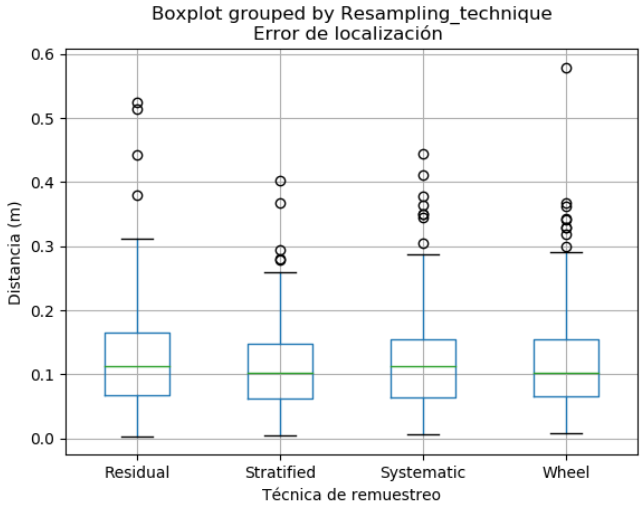


Figura 5.4. Error de localización en posición en función de la técnica de remuestreo.

de partículas del segundo clúster de mayor tamaño, puesto que tal como se observa en la Figura 5.8, la relación entre los tamaños de los dos clústeres de mayor tamaño es muy similar para las 3 técnicas de remuestreo.

Este detalle, que puede parecer rebuscado, puede utilizarse para aumentar la confianza en que el clúster elegido es el que representa correctamente la posición del robot. Esto se debe a que si el algoritmo se encuentra en la iteración final según el criterio de finalización original, si a este se le añade una comprobación de la varianza del segundo clúster más grande en comparación con la del primero, dicho criterio puede emplearse para determinar que aun no está claro cual de los dos clústeres representa la posición del robot y forzar a que el algoritmo continúe iterando.

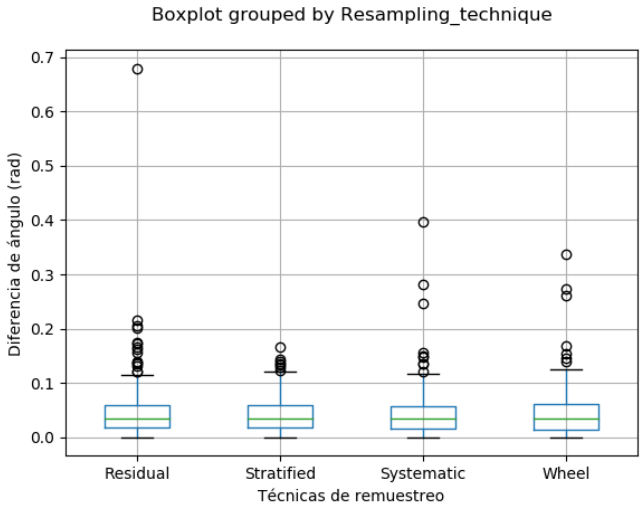
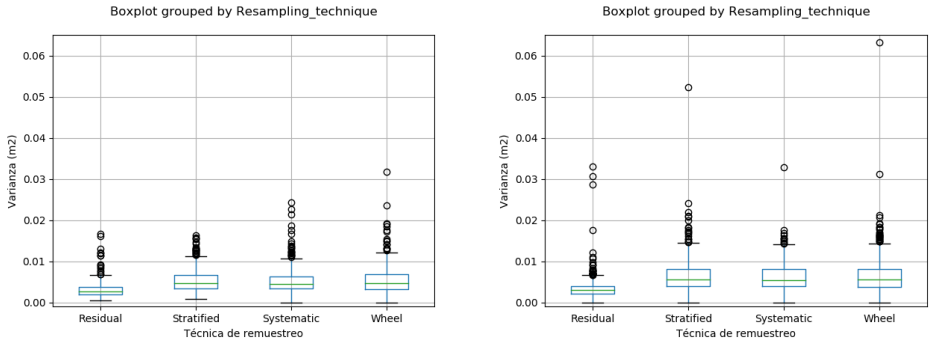


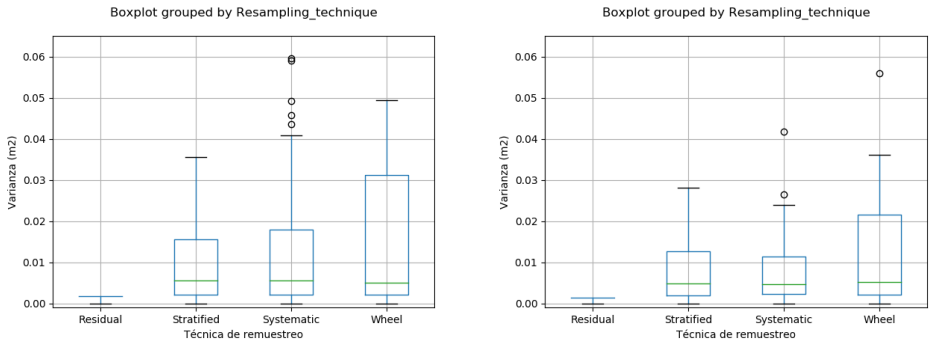
Figura 5.5. Error de localización en orientación en función de la técnica de remuestreo.



(a) Varianza en el eje X. **(b)** Varianza en el eje Y.

Figura 5.6. Varianzas en el clúster de mayor tamaño.

Esta pequeña diferencia es un punto a favor para el remuestreo de la rueda, aunque sigue sin ser suficiente para determinar que es claramente preferible uno de los algoritmos sobre los demás.



(a) Varianza en el eje X. **(b)** Varianza en el eje Y.

Figura 5.7. Varianzas en el segundo clúster de mayor tamaño.

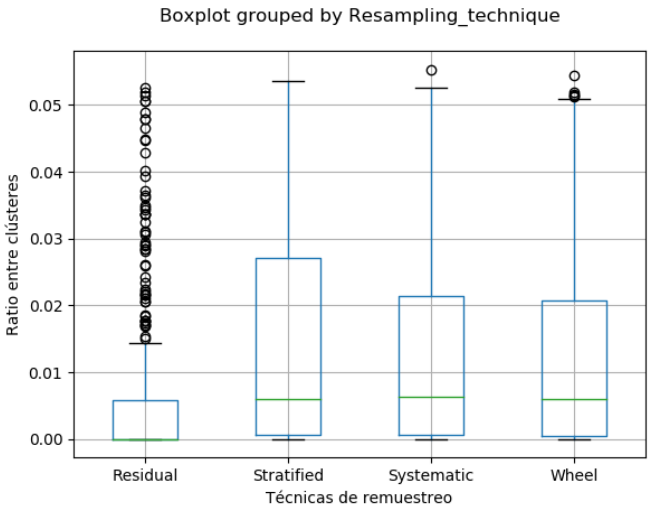


Figura 5.8. Cociente entre el número de partículas de los dos clústeres de mayor tamaño en el momento de la localización.

6

Conclusiones y futuros desarrollos

6.1. Conclusiones

Tras comparar los algoritmos de remuestreo bajo exactamente las mismas circunstancias se observa como, aunque no hay un claro ganador, sí hay un claro perdedor. El remuestreo residual es claramente inferior a los demás en la mayoría de aplicaciones debido a su considerablemente menor índice de éxito. Los demás algoritmos proporcionan resultados muy similares en todos los aspectos. De ser necesario elegir uno como el mejor, sería el remuestreo sistemático, que tiene el índice de éxito más elevado aunque por una diferencia pequeña (Tabla 6.1). Aunque hay un pequeño punto a favor del remuestreo de la rueda, hay que considerar que algunos de los parámetros del algoritmo de comportamiento del robot han sido elegidos experimentalmente para optimizar los resultados utilizando este algoritmo de remuestreo, lo que supone un punto en contra que decanta la balanza a favor del remuestreo sistemático.

Dicho esto, la explicación de por qué no hay un ganador no parece deberse únicamente a los algoritmos de remuestreo en sí, sino al resto de componentes del algoritmo. En este análisis comparativo se trata un problema real (aunque se haga a través de un simulador) y por ello hay muchas partes móviles, muchos componentes distintos que interactúan entre ellos de formas que no son completamente intuitivas ni expresables analíticamente. Por ello, los 3 algoritmos de remuestreo parecen encontrarse cerca de una aparente cota superior de índice de éxito que se debe a las imperfecciones en las otras componentes; principalmente la navegación del robot, que es un algoritmo bastante complejo. Esto implica que, siempre y cuando se emplee uno de los 3 algoritmos mencionados, es complicado que la calidad del remuestreo suponga una limitación considerable al rendimiento del robot.

Además, es muy notable el impacto que pueden llegar a tener pequeños ajustes en algunos parámetros como el valor de *alfa*, por ejemplo, para el cual por cuestiones de tiempo no se ha hecho un elevado número de pruebas para determinar su valor óptimo. Esto da pie a pensar que, si se dispusiese de cantidades de computación y de tiempo muy elevadas que permitieran ajustar los valores de todos y cada uno de los parámetros existentes en el filtro de partículas, sería posible mejorar los resultados sustancialmente hasta llegar a la cota superior marcada por la velocidad de movimiento del robot y su capacidad para recabar nueva información relevante

del entorno.

Tabla 6.1. Índice de éxito de cada algoritmo de remuestreo.

Remuestreo	índice de éxito (%)
Residual	64,0
Estratificado	80,0
Sistemático	82,4
De la rueda	81,6

6.2. Futuros desarrollos

En este apartado se proponen algunas líneas de desarrollo como continuación o variantes del proyecto planteado.

6.2.1. Análisis de la densidad de partículas

Aunque en este experimento, el número de partículas se ha elegido experimentalmente, para poder desplegar el robot a cualquier entorno sin necesidad de tener que replicar esta experimentación es necesario hacer consideraciones analíticas. De forma general es necesario evaluar la densidad de partículas en el entorno donde se mueve el robot. Esto se corresponde con la idea intuitiva de que si el entorno es el doble de grande, es decir, tiene el doble de superficie en donde el robot puede encontrarse, son necesarias el doble de partículas para la localización. En sentido estricto, esto es aplicable de forma directa solamente a entornos generados aleatoriamente y suficientemente grandes (esto último para asegurar que el entorno evaluado no se trata de una anomalía estadística). En caso contrario, es necesario considerar las características del entorno, es decir, si no se ha generado aleatoriamente cuáles son las consideraciones/restricciones que se han tenido en cuenta al generarlo.

Otra variable a tener en cuenta es el número de dimensiones a considerar, puesto que el número de partículas necesarias aumentará según aumenten estas. En particular, es necesario conseguir una cierta densidad de partículas en el entorno en cada una de las dimensiones a considerar, por lo que si se pretende replicar este mismo experimento a la navegación con drones el número de partículas necesario es mucho mayor, puesto que en vez de considerar posición en dos ejes y orientación en uno, se deben considerar posición y orientación en tres ejes. La densidad requerida en cada dimensión dependerá principalmente de las características del robot (precisión de sus sensores y actuadores y sus características dinámicas) y de las necesidades de la aplicación. El entorno tiene relevancia si altera como interactúan estas componentes, por ejemplo, si las superficies de las paredes reducen la precisión de los sensores, si el suelo provoca un cierto deslizamiento de las ruedas, etc. Pero si estas variables están controladas, se puede determinar cuál es la densidad de partículas necesaria en cada una de las dimensiones. En el caso de las dimensiones espaciales será conveniente que la distancia promedio esperada entre la posición del robot y las N partículas más cercanas sea menor que un cierto umbral. Conociendo la superficie en la que el robot puede operar el número de partículas necesario se puede calcular analíticamente. De la misma manera es necesario que la diferencia entre el ángulo inicial del

robot y el de las partículas de alrededor sea menor que un cierto valor para que las partículas puedan recuperarse de ese error inicial.

6.2.2. Adaptación del número de partículas

Esta es una de las variaciones más prometedoras del filtro de partículas, pues implementada correctamente permite reducir el tiempo de ejecución de forma considerable. En este proyecto se ha implementado la adaptación en base a umbrales de probabilidad mencionada en la Sección 2.1, pero con resultados muy inestables e impredecibles y por lo tanto inaceptables para el remuestreo. Esta modificación tiene implicaciones complejas, pues las variaciones en el número de partículas pueden depender tanto de las características topológicas como de las características métricas de la muestra de partículas (es decir, tanto por la posición estimada dentro del mapa por cada partícula como por el posible error respecto de esa posición).

Sin embargo, una modificación que posiblemente permitiría hacer más predecible este algoritmo es combinarlo con una técnica de *roughening*. Este tipo de técnicas introducen ruido en diferentes componentes del algoritmo para hacerlo más robusto y combatir la degeneración de partículas. Varios ejemplos de técnicas de este tipo empleadas en filtros de partículas pueden encontrarse en [15].

Otra posibilidad es implementar y emplear el muestreo KLD comentado en la Sección 2.1. Una alternativa que puede resultar más sencilla, puesto que su implementación es relativamente sencilla, es condicionar la variación del número de partículas a los resultados de la clusterización en cada iteración de esta. Por ejemplo, si se da la situación en que el 99 % de las partículas están concentradas en dos clústeres se podría reducir drásticamente el número de partículas para acelerar la computación sin perder apenas fiabilidad en el resultados. Es más, si en la reducción de partículas se reduce solo el número de partículas presentes en los clústeres más grandes, esta sería una forma de combatir la degeneración de partículas.

6.2.3. Criterios de finalización

Acorde con lo comentado en Capítulo 5 respecto de la varianza de los clústeres en el muestreo de la rueda, el criterio de finalización empleado puede ser de muchos tipos o incluso una operación lógica cuyo resultado puede depender de multitud de condiciones y restricciones. En este proyecto se han empleado umbrales de tamaño del clúster más grande y el segundo más grande, ajustando los umbrales experimentalmente. Se trata de un criterio sencillo y se puede decir que, en general, ha dado buenos resultados. Sin embargo, el uso de criterios más sofisticados como tests de hipótesis combinados con umbrales experimentales para otras variables podría suponer una gran mejora, evitando una convergencia prematura en algunas situaciones donde varias localizaciones son todavía posibles. Otra opción interesante es condicionar la finalización considerando todo el recorrido que se cree que ha realizado el robot desde que empezó a intentar localizarse, puesto que aunque esto es parecido a lo que hace el filtro de partículas, este *olvida* la información que ha recabado en el pasado y de esta queda solo el efecto que haya tenido en las partículas (que está sujeto a variaciones estadísticas por la propia naturaleza del algoritmo).

6.2.4. SLAM

Como se ha explicado en la Sección 1.3.3, el *SLAM* es un problema similar al de la localización global pero con complejidades añadidas. Sin embargo, tiene la ventaja de que no requiere de

un conocimiento inicial del entorno y es un algoritmo con el potencial de afrontar una mayor variedad de problemas, como el de la localización dinámica. Esto ha causado que haya cobrado una mayor importancia dentro de la robótica en los últimos años y que se hayan desarrollado variantes respecto del SLAM tradicional, por lo que sería una línea de continuación de este trabajo muy prometedora [3] [16].

A

Reducción del tiempo de simulación

A.1. Análisis del algoritmo

El filtro de partículas, como todo proceso de Montecarlo es un algoritmo inherentemente costoso computacionalmente, puesto que implica replicar un proceso muchas veces. Por esta razón, es importante reducir lo máximo posible el tiempo de computación de todas aquellas componentes del algoritmo que sean de orden $O(n)$ o superior (siendo n el número de partículas). Así pues, como aquello que no se puede medir no se puede mejorar, se ha modularizado el algoritmo en diferentes partes para medir los tiempos de computación de cada una por separado.

Aquellas partes cuyo tiempo de ejecución no depende del número de partículas tienen unos tiempos de ejecución varios órdenes de magnitud inferiores a las demás para los números de partículas que se emplean en este análisis. Es por ello, que dichas partes se ignoran en la optimización del algoritmo y se consideran solamente aquellas que sí dependen del número de partículas. Estas partes son 5: inicialización de las partículas, cálculo de los pesos, remuestreo, actualización del estado y clusterización.

La inicialización de las partículas no es relevante en este caso por tres razones: la primera es que en este experimento sólo es necesario realizarla una vez por cada trayecto del robot, la segunda es que el número de partículas que se maneja en este experimento es relativamente bajo (inferior a 5000 en todos los casos) y la tercera es que no es especialmente pesada puesto que solo implica la generación de una posición aleatoria dentro del mapa y de una orientación aleatoria de entre las 4 especificadas previamente. Dado que la generación de números aleatorios implementada es atómica esto no supone un coste relevante. En aquellas situaciones en que esto pudiera ser relevante (por ejemplo, si el mapa tuviese unas dimensiones muy grandes y fuera necesario generar muchas partículas o si se inicializasen partículas constantemente, como en aquellas variantes del algoritmo ideadas para ser robustas frente al secuestro del robot que se han mencionado), este coste podría reducirse empleando el mismo número aleatorio para generar la posición y la orientación de cada partícula.

El cálculo de los pesos, remuestreo y actualización del estado suponían inicialmente más del 95 % del tiempo de computación del algoritmo y es por ello que ha sido el foco de atención en

la reducción del tiempo de ejecución. La optimización de estas componentes ha supuesto una reducción del tiempo de ejecución del algoritmo de aproximadamente un orden magnitud, esto es, unas diez veces menos.

Para el criterio de finalización, se ha utilizado clusterización aglomerativa. Esta componente del algoritmo no se ha modificado puesto que no suponía una parte suficientemente relevante del tiempo de computación. Sin embargo, cabe destacar que esta componente no es $O(n)$ sino $O(n^2)$ o superior, lo que implica que para números más elevados de partículas, esto podría en efecto suponer un problema. Por ello, esta componente se beneficia especialmente de la dimensionalidad reducida del problema.

B

Alineación con los objetivos de desarrollo sostenible

B.1. Objetivo 8: Crecimiento económico

Descripción: un crecimiento económico inclusivo y sostenido puede impulsar el progreso, crear empleos decentes para todos y mejorar los estándares de vida.

Mejora: el uso de robots para sustituir o complementar a los humanos en tareas físicamente intensas o peligrosas conlleva una mejora de la economía, pues posibilita que los humanos se dediquen a tareas donde realmente aportan valor sin tener que preocuparse por otras que no lo aportan. Además, aumenta la seguridad y e bienestar en el trabajo, reduciendo los accidentes laborales tanto en cantidad como en gravedad.

B.2. Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación

Descripción: la industrialización inclusiva y sostenible, junto con la innovación y la infraestructura, pueden dar rienda suelta a las fuerzas económicas dinámicas y competitivas que generan el empleo y los ingresos. Estas desempeñan un papel clave a la hora de introducir y promover nuevas tecnologías, facilitar el comercio internacional y permitir el uso eficiente de los recursos. Sin embargo, todavía queda un largo camino que recorrer para que el mundo pueda aprovechar al máximo este potencial. En especial, los países menos desarrollados necesitan acelerar el desarrollo de sus sectores manufactureros si desean conseguir la meta de 2030 y aumentar la inversión en investigación e innovación científicas. La innovación y el progreso tecnológico son claves para descubrir soluciones duraderas para los desafíos económicos y medioambientales, como el aumento de la eficiencia energética y de recursos. A nivel mundial, la inversión en investigación y desarrollo (I+D), como porcentaje del PIB, aumentó de un 1,5 % en el 2000 a un 1,7 % en el 2015, y continuó casi en el mismo nivel en el 2017. Sin embargo, en las regiones en desarrollo fue inferior al 1 %. En términos de infraestructura de comunicaciones, más de la mitad de la población mundial está ahora conectada y casi toda la población global

vive en un área con cobertura de red móvil. Se estima que, en 2019, el 96,5 % de la población tenía cobertura de red, como mínimo, 2G.

Mejora: la innovación en la industria asociada al uso de robots móviles permite, en efecto, potenciar las infraestructuras y aumentar la eficiencia de los procesos.

B.3. Objetivo 13: Acción por el clima

Descripción: el 2019 fue el segundo año más caluroso de todos los tiempos y marcó el final de la década más calurosa (2010-2019) que se haya registrado jamás. Los niveles de dióxido de carbono (CO2) y de otros gases de efecto invernadero en la atmósfera aumentaron hasta niveles récord en 2019. El cambio climático está afectando a todos los países de todos los continentes. Está alterando las economías nacionales y afectando a distintas vidas. Los sistemas meteorológicos están cambiando, los niveles del mar están subiendo y los fenómenos meteorológicos son cada vez más extremos. El Acuerdo de París, aprobado en 2015, aspira a reforzar la respuesta mundial a la amenaza del cambio climático manteniendo el aumento global de la temperatura durante este siglo muy por debajo de 2 grados Celsius con respecto a los niveles preindustriales. El acuerdo también aspira a reforzar la capacidad de los países para lidiar con los efectos del cambio climático mediante flujos financieros apropiados, un nuevo marco tecnológico y un marco de desarrollo de la capacidad mejorado.

Mejora: los robots móviles son fundamentalmente eléctricos y sustituyen en muchos casos a vehículos alimentados por combustibles fósiles, por lo que la propagación de su uso implica una reducción directa de la contaminación medioambiental. Además, requieren en general menos energía para realizar las mismas tareas.

Bibliografía

[1] I. Bukhori and Z. H. 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, no. 4, p. 172988141771746, 2017.

[2] I. Bukhori, Z. H. Ismail, and T. Namerikawa, "Detection strategy for kidnapped robot problem in landmark-based map monte carlo localization," in *2015 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pp. 75–80, IEEE, 18/10/2015 - 20/10/2015.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[4] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Intelligent robotics and autonomous agents, Cambridge Mass.: MIT Press, 2005.

[5] T. B. Schon, R. Karlsson, and F. Gustafsson, "The marginalized particle filter in practice," in *2006 IEEE Aerospace Conference*, pp. 1–11, IEEE, 04-11 March 2006.

[6] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *The International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.

[7] T. Schon, F. Gustafsson, and P.-J. Nordlund, "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2279–2289, 2005.

[8] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, pp. 1225–1232, IEEE, 24-28 April 2000.

[9] T. Li, M. Bolic, and P. M. Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70–86, 2015.

[10] J. D. Hol, T. B. Schon, and F. Gustafsson, "On resampling algorithms for particle filters," in *2006 IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 79–82, IEEE, 13/09/2006 - 15/09/2006.

[11] W. Adiprawita, A. Suwandi Ahmad, J. Sembiring, and B. R. Trilaksono, "A novel resampling method for particle filter for mobile robot localization," *International Journal on Electrical Engineering and Informatics*, vol. 3, no. 2, pp. 165–177, 2011.

[12] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2442–2450, 2005.

[13] B. Balasingam, M. Bolic, P. M. Djuric, and J. Miguez, "Efficient distributed resampling for particle filters," in *Efficient distributed resampling for particle filters* (Balakumar Balasingam, Miodrag Bolić, Petar M. Djurić, and Joaquín Míguez, eds.), pp. 3772–3775, IEEE, 22/05/2011 - 27/05/2011.

[14] S. I. Martínez, J. A. C. Rocha, J. L. Menchaca, M. G. T. Berrones, J. G. Obando, J. P. Cobos, and E. C. Rocha, "An autonomous navigation methodology for a pioneer 3dx robot," *Computer Technology and Application*, vol. 5, no. 2, 2014.

[15] Tiancheng Li, Tariq P. Sattar and S. S. Qing Han, *Roughening Methods to Prevent Sample Impoverishment in the Particle PHD Filter: 9-12 July 2013, Istanbul, Turkey*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, 2013.

[16] M. Lin, C. Yang, D. Li, and G. Zhou, "Intelligent filter-based slam for mobile robots with improved localization performance," *IEEE Access*, vol. 7, pp. 113284–113297, 2019.