Comparative Analyses of Mobile Robots Localization Algorithms based on Particle Filters

Author: Julio Labora Gómez
Universidad Pontificia Comillas
Madrid, Spain
julio.labora.gomez@alu.comillas.edu

Director: Jaime Boal Martín-Larrauri
Universidad Pontificia Comillas
Madrid, Spain
jaime.boal@iit.comillas.edu

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.**

Resumen—**La localización de robots móviles es clave en multitud de apliaciones 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 disintas 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.**

Index Terms—**autonomous mobile robots, particle filter, Bayes filter, robot localization, resampling techniques, state estimator, Industry 4.0**

I. 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 process. 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.

Therefore particle filters are very useful in Industry 4.0 applications, in which mobile robots are an increasing trend. One example is the renowned Seat car factory in Martorell, that has substituted its loading trucks for AGV's in the transport of parts. These AGV's perform SLAM navigation and are fully electric. They do not follow marks on the ground and do not run on rails, they are able to move freely within the factory, which avoids maintenance costs and civil work [5]. Another similar application, but in the retail sector, is the use of mobile robots for the transport of packages in warehouses. Amazon is a pioneer in this application [6]. Other popular applications include medical and surgical issues, personal assistance and security, as well as ocean and space exploration. For example, mobile robots are used in nuclear power plants to reach areas which are inaccessible for humans. Maintenance is also another field in which mobile robots are an increasing trend, as they simplify some of the tasks required for the maintenance of big infrastructures, such as ships or planes. They are also popular in exploring and monitoring tasks, whether it is for security and patrolling, thermal monitoring (which is crucial in server farms) or exploration of mines [7].

II. 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) [8].

However, several modifications of the basic algorithm exist, so that it can better handle specific problems such as those mentioned in Section I, 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.

Possibly the most well-known variation is the Rao-Blackwellized particle filter (also known as marginalized particle filter). This algorithm exploits a feature that some dynamic systems have, in which the state space is non-linear but it contains a linear subspace. In these cases, the linear subspace can be handled separately using a Kalman filter, which is an analytical state estimator. Because it is analytical, the Kalman filter is much more computationally efficient than the particle filter, so this modification allows a faster convergence. Additionally, the Kalman filter is an optimal state estimator (in those cases in which it can be used), so the state estimation of the marginalized particle filter is better than estimation of the regular particle filter [9] [10].

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.

One strategy is to use the sum of the weights of the particles to determine if enough samples had been drawn of more are needed. To do this, a preset threshold is used, and when the sum of the weights reaches this threshold, no more particles are drawn. The correct choice of this threshold is key for the success of the strategy [11].

Another strategy is to use KLD-sampling (where KLD stands for *Kullback-Leibler divergence*, which is a generally accepted technique to measure the difference between two distributions, despite that technically it is not a metric, because although its value is non-negative and is only 0 if both distributions are identical, but it does not have the symmetric nor the triangular properties). This technique aims to bound the estimation error caused by the representation of the particle filter probability density distribution using samples. To calculate this error, the Kullback-Leibler divergence of the difference between the sample based probability density function and the believe of the true probability density function is used (Equation 1). This technique can be implemented in particle filters in order to ensure that the number of particles is enough so that the following statement holds: *with a probability of* $1-\delta$*, the approximation error between both distributions is smaller than* ϵ*, being* δ *and* ϵ *preset values.* To achieve this, particles are drawn from the probability density function until the difference between the sample based distribution and the believed true distribution is smaller than the preset threshold. This strategy has the same advantage as the previous one, as it enables to keep just the necessary number of particles so that unnecessary computation is avoided [11].

$$K(p,q) = \sum_{x} p(x) log \frac{p(x)}{q(x)} \tag{1}$$

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 [12].

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 [13] [14] [15]. Popular modern variations include residual systematic, modified and distributed resampling [16] [17].

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, though the generation of N random numbers is still costly. This technique guarantees that all particles with normalized weight greater than $2/N$ are drawn, as those particles necessarily cover one full subrange.

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 2.

$$u_n = u_1 + \frac{n-1}{N} \tag{2}$$

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 3. For this selection, any other resampling method can be used.

$$\hat{w_i} = w_i - \frac{n_i}{N} \tag{3}$$

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 4, in which the weight is calculated as a function of the probability associated to that particle. Non-modified resampling techniques follow the exact same equation with $\alpha = 1$.

$$\omega_i = (p_i)^{\alpha} \tag{4}$$

Algorithm 1: Wheel resampling algorithm

```
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
```

III. Problem statement

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. The components already have their required parameters fine-tuned so that the robot behaves in a fairly smart way. One downside, is that the fine-tuning of those parameters has been done with the wheel resampling algorithm, so this may be a source of bias in this comparison.

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.

The reduction of the computation time has been crucial for the analyses, because of the high computation cost of the particle filter. In fact, this analysis could not have been possible without this reduction, as the total simulation time that has been required is approximately ten days just for the generation of the final data (prior simulations have been necessary to debug the algorithms), and the computation time reduction achieved is approximately one order of magnitude (i.e. 10 times less). It would have been necessary over a 100 days to generate this amount of data. In this matter, the automation of the simulations has also been paramount to the success of the project, as the computer has been able to keep running the simulations overnight, both for debugging and for data generation.

In order to show the resampling algorithms at their best, those of these parameters which are believed to be more determining in the performance of the algorithms are furtherly fine-tuned to improve the results. These parameters are: *alpha* and the number of particles. Additionally, multiple strategies to adapt de number of particles have also been tested. As it is not possible to simulate all the combinations of the chosen parameters/variations for each of the four resampling algorithms in a reasonable amount of time, a prior comparative analysis is done to decide which is the most adequate variation to perform the final comparison. This comparative analysis has been carried out using the multinomial resampling algorithm to avoid introducing bias in the analysis.

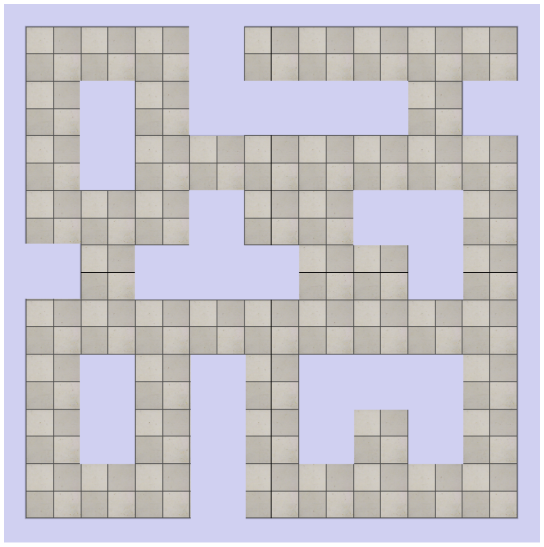

Fig. 1. Robot environment. It is 9 m x 9 m, consisting of 81 1 m x 1 m cells.

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). The 0.65 meters threshold corresponds to an experimental value, equal to a parameter used in the localization criterion, which is explained below. The 45 degrees threshold is considered to be approximately the maximum difference from which the particle filter can recover after the localization. Additionally, to avoid excessive computation an expedition is considered invalid (and therefore not taken into account in any way and simply repeated) if the number of steps of the navigation algorithm of the robot is higher than 500, which are more than enough for the robot to localize itself under normal circumstances.

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 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 [18]. 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.

Other parameters which are relevant, but are not modified for this experiment are:

- Sensors and actuators sampling time: 0.05 seconds
- Particle filter sampling time, and therefore resampling and finalization criterion sampling time: 1 second
- Sonar sensors noise: 0.6 m
- v and w measurement noise: 0,15 m/s and 0,2 rad/s respectively

Additionally, the navigation algorithm and the localization criterion are also relevant. The first one is highly complex with many parameters, but it consists of an state machine that evaluates the environment (i.e. where are the walls) and a wall follower algorithm implemented with two PID controls in cascade configuration. The controlled variables are the relative angle between the wall and the robot, and the lateral distance to the wall. Regarding the localization criterion, agglomerative clustering is used [19], with average distance being used as distance between groups and a threshold of 0.65 m beyond which, clusters are not merged. The clustering is performed considering only the euclidean distance and not taking into account the orientation of the particles. 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.

IV. Parameter analysis

To reduce the number of required simulations for the final comparison, some parameters have been prefixed. These 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 a constant 5% decrease each iteration of the particle filter. Adaption of the number of particles using a threshold for the sum of the weights has been implemented, but with very poor results, so it has not been considered for the final comparison.

As explained above, 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 number of expeditions to be performed with each of the alpha values is 25 with a constant number of particles of 2000, which may seem small numbers, but the total simulation time required has been approximately 12 hours.

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. This is consistent with the rationale explained in Section II, that lower values of alpha reduce particle degeneracy, and compensate for the high variance of multinomial resampling.

The localization error, on the contrary, does not show a clear trend with respect to *alpha*, although the overall values are quite good (less than 0.2 meters error in an environment of over 80 square meters).

Taking into account the results shown in both figures, 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 information about the experiments and the final choices are.

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

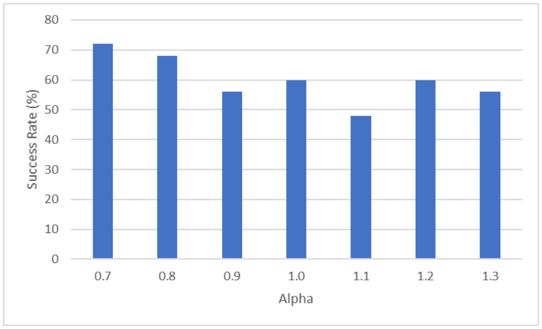

Fig. 2. Localization success rate grouped by alpha.

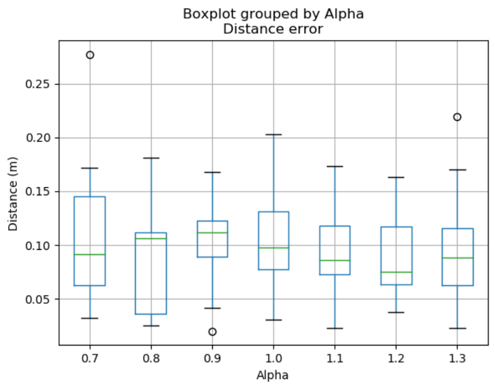

Fig. 3. Localization distance error grouped by alpha.

The number of particles that have been analyzed are 1500 to 4000 in steps of 500. For each of them, 60 expeditions have been performed with a value of *alpha* equal to 1. These simulations have taken approximately 42 hours. The chosen value is 2500.

For each of the adaption strategies 120 expeditions have been simulated, with an initial particle number of 2000 and *alpha* equal to 1. The required simulations have taken approximately 16 hours. The strategy which has proven to be better is periodically decreasing the number of particles, which also reduces the computation time considerably.

TABLE I
Parameter summary.

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

V. Comparative analysis

In this section, 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 explained in Section III). The total simulation time has been approximately 36 hours per resampling algorithm, about 6 days in total.

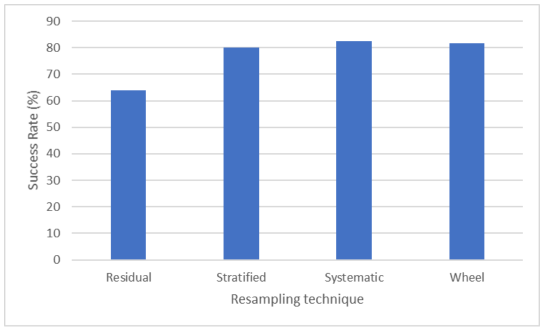

Fig. 4. Success rate grouped by resampling technique.

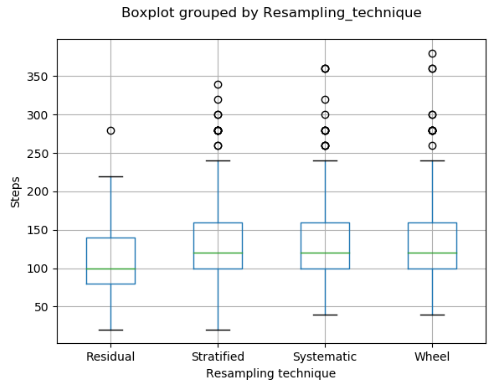

Fig. 5. Number of steps before localization grouped by resampling technique.

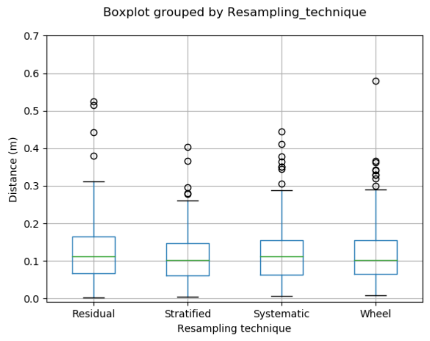

Fig. 6. Localization distance error grouped by resampling technique.

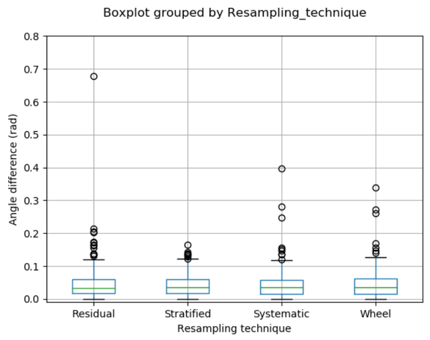

Fig. 7. Localization orientation error grouped by resampling technique.

Observing the Figure 4, 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 5 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.

Figures 6 and 7 show 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 algorithm.

VI. Conclusions and Future Developments

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. An algorithm that reduces the particles only when they are strictly unnecessary would

TABLE II
Success rate of each resampling algorithm.

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

most definitely make a difference.

Another source of improvement is the sophistication of the finalization criterion. Although, the one explained previously has proven to be reasonably good, a criterion that takes into account more information, regardless if it is information on the environment, the resampling technique or any other, can also make a significant difference in the localization results.

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] David Galán "Así son los robots eléctricos y autónomos de SEAT Martorell que han jubilado a los camiones para el transporte de piezas" https://www.motorpasion.com/seat/asi-robots-electricos-autonomos-seat-martorell-que-han-jubilado-a-camiones-para-transporte-piezas

[6] Matt Simon "Inside the Amazon warehouse there humas and machines become one" https://www.wired.com/story/amazon-warehouse-robots/

[7] Bennett Brumson "New applications for mobile robots" https://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/New-Applications-for-Mobile-Robots/content_id/3362

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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