Particle Swarm Localization for Mobile Robots Using a 2D Laser Sensor

João Luiz C. Carvalho[1,2], Paulo César M. A. Farias[1], Edmar Egidio P. de Souza[1], Eduardo F. de Simas Filho[1]
[1]*Digital Systems Laboratory, Electrical Engineering Program*
Federal University of Bahia
Salvador, Brazil
[2]*Center for Science and Technology in Energy and Sustainability*
Federal University of The Recôncavo da Bahia
Feira de Santana, Brazil
joao.luiz@ufrb.edu.br, {paulo.farias, edmar.egidio, eduardo.simas}@ufba.br

***Abstract*—Mobile robot localization is a complex task, specially in unstructured indoor environments, due to noise and wrong data association from sensors. The localization procedure is even harder when the vehicle has low confidence about its last pose estimate, situation that requires a Global Localization procedure. In this work, a Global Localization algorithm based on Particle Swarm Optimization (PSO) is integrated with a Pose Tracking algorithm, the Perfect Match, to obtain a robust localization technique. Results show that this approach can solve the problem of Global Localization with good performance.**

***Index Terms*—Mobile Robot; Global Localization; Particle Swarm Optimization; Map Matching**

I. Introduction

In virtually all mobile robotics applications, localization plays a key role. To move around and perform tasks safely and efficiently, the mobile terrestrial robot must know its own position (x, y coordinates) and orientation (θ) in relation to the environment (the world coordinate system) with low degree of uncertainty.

In some situations the robots localize themselves only using proprioceptive sensors that measure system's internal values, an approach known as dead reckoning. On the other hand, map based localization techniques relies on the position matching between features extracted from the environment (using laser-based sensors or camera, for instance) and its respective coordinates on the map. Since the uncertainty in position grows without bound using dead-reckoning, the robustness of map-based algorithms is substantially higher than the former approach [1].

Most of the mobile robots rely on the perception of the environment to obtain their location by detecting features, like landmarks or contours, present in both the environment and the map. These elements are usually represented by points in the map coordinate system. Also, the robot can detect the relative position of these elements using perceptive sensors. Since the robot knows the coordinates of all features in the map (x^f, y^f) and also is able calculate its range and bearing from itself (d_x^f, d_y^f, ϕ^f), it is only required a linear transformation between the robot coordinate system and the map coordinate system to obtain the robot pose (x, y, θ) in relation to the map.

However, the coordinates of detected features are only approximate due to the noise and uncertainty of sensor measurements. Moreover, the correspondence between the observed feature and its true position is not always guaranteed, due to features similarities or the uncertainty from the sensing. The uncertainty about its own localization can be utmost when the vehicle has to localize itself from scratch, i.e., without any previous localization estimate, a process known as Global Localization. Thus, a robust localization algorithm must be able to handle common ambiguities and multiple hypotheses when determining a robot pose.

This work introduces a method to localize a mobile robot based on the contours of the 2D map, which is a common approach for unstructured indoor environments. The method is based on a Pose Tracking algorithm, described in section II, and a Global Localization algorithm based on PSO, described in section III. The section IV details the methodology used in the tests and section V discuss the experimental results obtained from a real Automated guided vehicle (AGV) in different tests. Finally, the section VI comprises the authors conclusions about this work.

II. Perfect Match Algorithm

The Perfect Match is a light computational pose tracking algorithm that uses a map matching approach. It was firstly introduced by M. Lauer, S. Lange, and M. Riedmiller [2] to localize a soccer playing robot on the game field with white markings using an omnidirectional color camera. H. Sobreira et al. [3] extended the use of Perfect Match to localize autonomous robots in a 2D occupancy grid map using laser sensors to detect the contours of the environment and M. Pinto et al. [4] adapted the Perfect Match from 2D to the 3D localization using 3D laser sensors.

The algorithm implemented in [3] incrementally tries to find the best fit of the vehicle on a 2D map based on its previous pose estimate and the N points produced by the laser sensor. The new pose estimate is computed by minimizing the fitting error E between the laser's point cloud and the occupied cells

2643-6264/19/$31.00 ©2019 IEEE
DOI 10.1109/BRACIS.2019.00057

of the map. Hence, the first step of the algorithm is to calculate E as defined by (1):

$$E = \sum_{i=0}^{N} \left(1 - \frac{L_c^2}{L_c^2 + d_i^2}\right), \quad (1)$$

where $d_i = (x_i, y_i)$ is the x and y-distances between the i^{th} point of the laser in the world coordinate system and the nearest occupied cell. The L_c, as described in [3], is a parameter that limits the influence of scan points too far from the map contours, which could indicate the presence of outliers. Since all x and y-distances from every cell to the nearest occupied one can be precomputed and stored in a pair of 2D matrices of distances, the computational cost for obtaining each d_i is reduced to a couple of array readings.

The second step is to optimize the estimate by minimizing E. The optimization is carried by the Resilient Backpropagation (RPROP) algorithm, which adjusts the parameters x, y and θ of the vehicle's pose estimate based on gradients extracted from the matrices of distances. Details of pose optimization using RPROP can be found in [2] and [4].

In order to the optimization procedure converge to a good estimate, the last known pose estimate must be near the ground truth pose. This is the reason the Perfect Match does not perform well without a previous estimate. If the previous estimate is too far, or even too rotated compared to the ground truth, there might not be enough optimization steps to find an acceptable pose estimate. Although some localization algorithms handle multiple pose hypotheses at the same time, the Perfect Match works with only one estimate. Hence, when a vehicle running the Perfect Match starts or gets lost, a different approach for localization is required.

III. Particle Swarm Optimization

Kennedy and Eberhart [5] introduced the Particle Swarm concept for the optimization of nonlinear functions. The nature-inspired meta-heuristic roughly consists of moving a set of N candidate solutions (the swarm of particles) around the D-dimensional search space towards the region of best solutions found. The algorithm keep the best fitness value of each particle obtained so far (personal best, or $\mathbf{p}_{best}$) and the best score of the entire swarm obtained so far (global best, or $\mathbf{g}_{best}$). The simple form of the algorithm updates, at each iteration k, the position $p_i \in \mathbb{R}^D$ and speed $\mathbf{v}_i \in \mathbb{R}^D$ of each particle i using (2) and (3):

$$\mathbf{p}_{i(k)} = \mathbf{p}_{i(k-1)} + \mathbf{v}_{i(k-1)} \quad (2)$$

$$\mathbf{v}_{i(k)} = \mathbf{v}_{i(k-1)} + \phi_1 r_{1(k)} (\mathbf{p}_{best} - \mathbf{p}_{i(k)}) + \phi_2 r_{2(k)} (\mathbf{g}_{best} - \mathbf{p}_{i(k)}), \quad (3)$$

where ϕ_1 and ϕ_2 are acceleration constants and $r_{1(k)}$ and $r_{2(k)}$ are pseudo-random numbers sampled from a uniform distribution $U(0, 1)$.

The fitness function f receives a particle's position p_i inside the search space and returns a value f_i to that particle. The PSO algorithm uses this value to update the $p_{best,i}$ and $\mathbf{g}_{best}$ positions, but makes no assumption about the implementation details of fitness function. The PSO algorithm is summarized in the Algorithm 1.

Algorithm 1: Particle Swarm Optimization

```
Result: Global Best Solution Approximate
k = 0;
for each i ∈ [1, N] do
    f_i(k) = p_i(k) = v_i(k) = 0;
end
while k < Max. number of iterations do
    for each i ∈ [1, N] do
        f_i(k) = f(p_i(k));
        if f_i(k) < p_best,i(k) then
            p_best,i(k) = f_i(k);
        end
    end
    g_best(k) = min(p_best,j(k)), j ∈ [1, N];
    k++;
    for each i ∈ [1, N] do
        Update v_i(k) and p_i(k) using (3) and (2);
    end
end
```

A. Particle Swarm Localization

In the context of mobile localization, the PSO algorithm can be used to estimate a vehicle's pose in a 2D or 3D space, and each pose hypothesis is considered a candidate solution, or a particle. Q. Zhang et al. [6] implemented the PSO to globally localize mobile robots with a few iterations along with a scan-matching technique. A. Vahdat, N. NourAshrafoddin and S. Ghidary [7] demonstrated that the Global Localization using PSO yields a good performance for a two wheel robot equipped with a 360 degrees laser scan, in comparison with other methods, namely Differential Evolution (DE) and Monte Carlo Localization (MCL). A. Pinto, A. Moreira and P. Costa [8] also proposed the use of PSO as complement of a Pose Tracking algorithm in situations when a small mobile robot gets lost.

In this work, a Global Localization technique based on PSO is integrated with the Perfect Match algorithm to obtain a robust localization technique. The Particle Swarm Global Localization (PSGL) estimates the robot's pose, a 3-vector $\mathbf{p}_G = (x, y, \theta)$, inside a 2D occupancy grid (the world map). The $\mathbf{p}_G$ is the $\mathbf{g}_{best}$ of the swarm after a certain number of iterations. To work together with Perfect Match, the PSGL must starts when the robot is turned on or when the map-matching error is high, meaning that it got lost.

A initial swarm is generated and randomly distributed over the free area of the map. In the following steps, each particle is updated using (2) and its velocity using (4):

$$\mathbf{v}_i = K \cdot (\mathbf{v}_{i-1} + \phi_1 \cdot r_1 \cdot (\mathbf{p}_{best,i} - \mathbf{p}_i) + \phi_2 \cdot r_2 \cdot (\mathbf{g}_{best} - \mathbf{p}_i)), \quad (4)$$

where ϕ_1 and ϕ_2 are empirical values for the social and cognitive factors of the swarm, respectively. In this work, both

ϕ_1 and ϕ_2 were set to 2.05. The K parameter is a constriction factor [9] given by (5):

$$K = \frac{2}{|2 - \psi - \sqrt{\psi^2 - 4\psi}|}, \quad \text{where} \begin{cases} \psi = \phi_1 + \phi_2, \\ \psi > 4. \end{cases} \tag{5}$$

In many real-life scenarios, the Global Localization is performed while the robot is moving. However, in the simple PSO algorithm, only the swarm moves around the search space, while each $\mathbf{p}_{best}$ keeps static or is replaced by a better candidate. This may cause the best solutions found, $\mathbf{p}_{best}$'s, to be outdated, since the vehicle keeps changing its pose and old solutions could make no sense anymore. Because of that, the $\mathbf{p}_{best}$s' poses need to be dynamically refreshed according to the robot movement.

Many vehicles running Robot Operating System (ROS) have an odometer system that publishes the linear v_k^o and angular ω_k^o velocities extracted from sensors like wheel encoders or Inertial Measurement Units (IMU). In this work, v_k^o and ω_k^o are used to update each $\mathbf{p}_{best}$ pose (x_k, y_k, θ_k) after the time interval between two consecutive iterations Δt_k, according to (6) for the velocity motion model from [10]:

$$\begin{pmatrix} x_k \\ y_k \\ \theta_k \end{pmatrix} = \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{pmatrix} + \begin{pmatrix} -\frac{v_k^o}{w_k^o}\sin\theta_{k-1} + \frac{v_k^o}{w_k^o}\sin\left(\theta_{k-1} + \omega_k \Delta t_k\right) \\ \frac{v_k^o}{w_k^o}\cos\theta_{k-1} - \frac{v_k^o}{w_k^o}\cos\left(\theta_{k-1} + \omega_k \Delta t_k\right) \\ \omega_k \Delta t_k \end{pmatrix}. \tag{6}$$

B. The Fitness Value

The fitness function used in PSGL is based on the map-matching error computed by the Perfect Match algorithm. However, the optimization step with RPROP is not executed because it would be applied to each particle individually, and the core property of a Particle Swarm algorithm is to optimize each candidate by interacting it with the entire swarm.

Eq. (1) specifies the fitness value computing procedure. The fitness function makes no assumptions about the vehicle's pose on the map, so particles moved to non-free cells can eventually assume a good fitness value. Because of that, the PSGL assigns a large value for particles on occupied cells, avoiding them to be considered as a valued candidate solution.

As mentioned earlier in this section, The $\mathbf{p}_{best}$'s change its poses according to the linear and angular velocities given by odometry, so they also need to update its fitness values. Considering this, PSGL was implemented so that the iteration rate matches the odometer velocities update rate, and each new step of the algorithm causes the fitness value of N swarm particles and N $\mathbf{p}_{best}$'s to be updated.

IV. Methodology

The Particle Swarm Localization was tested in the Husky A200 AGV equipped with a SICK LMS1xx 2D-LiDAR laser sensor with 270º of range. The environment is a robotics laboratory with approximately 90 m^2 occupied with tables, chairs, cabinets and two separate small rooms. The occupancy grid map with 2.5 cm of resolution (Fig. 1) was previously created by Hector SLAM [11] mapping software running on the Husky.

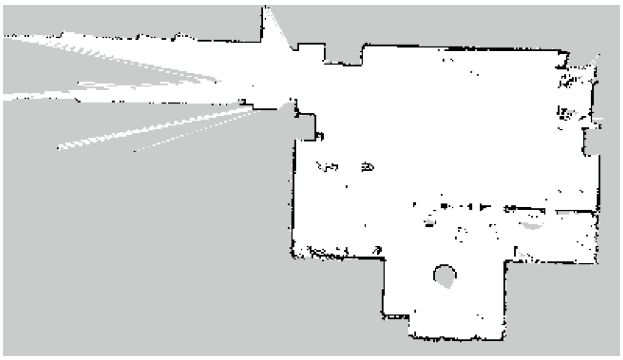

Fig. 1. Occupancy grid map: each pixel is a 6.25 cm^2 cell. White pixels are free cells and black pixels are occupied. Unexplored regions are in gray color.

The experiments used offline data acquired from a 16 minutes ride inside the laboratory. The algorithm evolves the swarm whenever a new odometry message is received. Hence, given that the Husky odometry system publishes its velocities in intervals of 100 ms, the rate of swarm evolution is 10 times per second. The algorithm performed well on a Intel Core i5 CPU with 8GB of RAM memory. Nevertheless, the swarm update rate can be lowered by skipping consecutive odometry messages, reducing the CPU usage.

Fig. 2 illustrates the testing system. The Perfect Match uses the map and the laser scan to refine the pose estimate, whereas the PSGL uses the map to generate the swarm and odometry information to update p_{bests}'s. PSGL calls the Perfect Match to obtain fitness values for each particle in the swarm whenever they change its the poses.

Additionally, as the Fig. 2 depicts, the performance of PSGL algorithm was compared to a typical Global Localization algorithm, the Adaptative Monte Carlo Localization (AMCL), which is well described in [10]. The AMCL is a probabilistic localization algorithm that uses a particle filter approach to estimate the pose of a robot against the map using odometry and laser scan information. An AMCL implementation is available in the ROS Framework.

V. Experimental Results

Figs. 3 and 4 show the Global Localization running with 1000 particles at 9 different iterations: $k = 1$, $k = 3$ and $k = 5$ in the upper line, $k = 10$, $k = 20$ and $k = 30$ in the mid line, $k = 50$, $k = 100$ and $k = 150$ in the bottom line). The dark blue arrow represents the pose estimated by PSGL

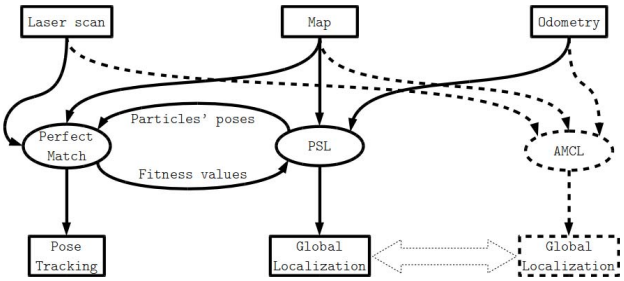

Fig. 2. Diagram of the testing system.

and the red arrow is the true pose. In most experiments the algorithm found the correct pose with less than 10 iterations. In the Fig. 3, the dark blue arrow matches the red arrow from the 10^{th} iteration onwards.

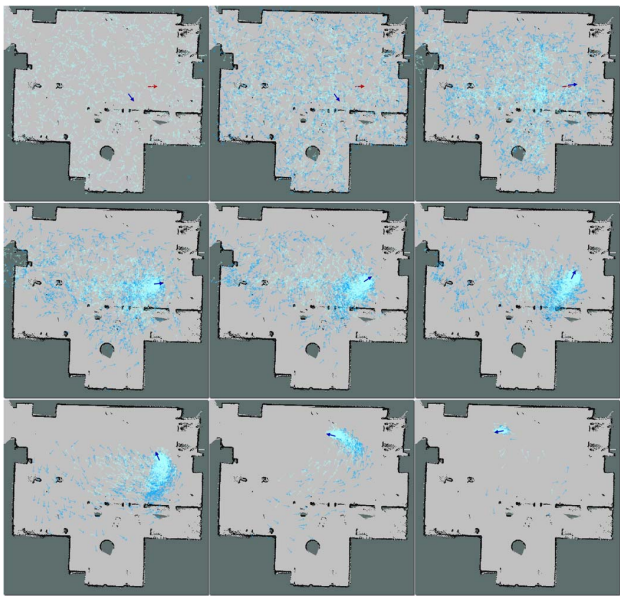

Fig. 3. From the upper left to the bottom right: iterations 1, 3, 5, 10, 20, 30, 50, 100 and 150 of the PSGL. The particle swarm is in light blue, and its P_{best}'s poses are in medium blue. The dark blue arrow is G_{best} pose and the ground truth pose is the red arrow.

A particular experiment (Fig. 4) was performed to test the robustness of PSGL algorithm when the contours of the environments doesn't quite match the contours of the occupancy grid. A 150x150 cm wooden board was placed at approximately 40 cm from the wall and the global localization started when the vehicle was placed near the board. As Fig. 4 shows, the PSGL also spent less than 10 iterations to find a good pose estimate. The laser scan points are showed in yellow and the board contours detected by the laser can be seen in the upper side of the map.

The performance of PSGL was compared to AMCL under similar conditions. In order to obtain a direct comparison, both localization algorithms, PSGL and AMCL, were started con-

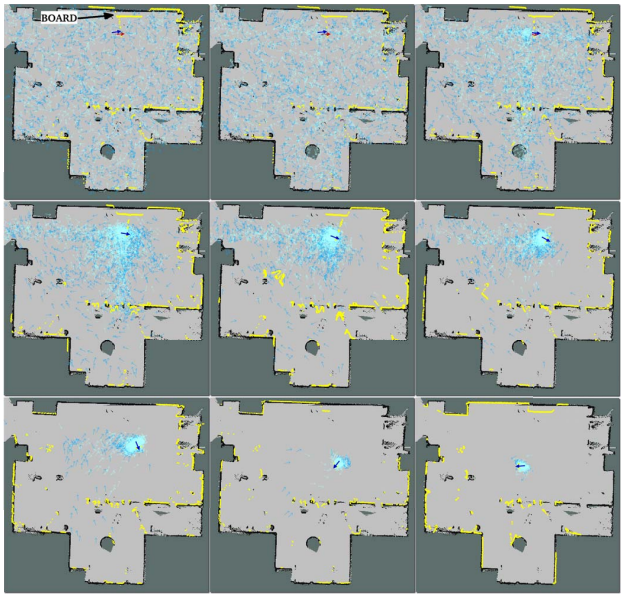

Fig. 4. Particle Swarm Localization in a scenario with unexpected contours.

currently with the same initial population size (1000 particles). However, due to its implementation details, the particles set of AMCL is updated only upon a minimum linear (d_{min}) or angular (a_{min}) vehicle displacement. So in a first test the d_{min} and a_{min} were set to 1 cm and 0.01 radians, respectively. The Euclidean distance between an estimate and the robot's ground truth, referred to as position estimate error, was captured in four different scenarios and showed in Fig. 5, whose y-axis is logarithmic for better visualization. The data collected span in 5 seconds of execution time, starting from the beginning of global localization, and comprise 50 iterations of PSGL.

The Fig. 5 shows that the PSGL converges to a good estimate before the AMCL, usually in the first second of Global Localization. However, both algorithms keep a similar performance as their particles set approximate the global minimum.

In another test, the AMCL was configured with the default minimal displacements: $d_{min} = 20$ cm and $a_{min} = 0.2$ radians. The results are showed in the Fig. 6, which reproduces the position estimate error, also in the logarithmic scale, for aproximately 25 seconds of global localization and 200 iterations of PSGL. It demonstrates that the PSGL has a better performance when AMCL have standard settings and same initial population size.

The PSGL was also tested for different population sizes. Table I shows different number of populations and the number of trials that succeeded – which means that the PSGL found a good pose estimate, and the number of failures – meaning that the algorithm could not find an acceptable estimate even after 200 iterations.

The size of the environment and the resolution of the map are sensitive factors for a localization system, so the number

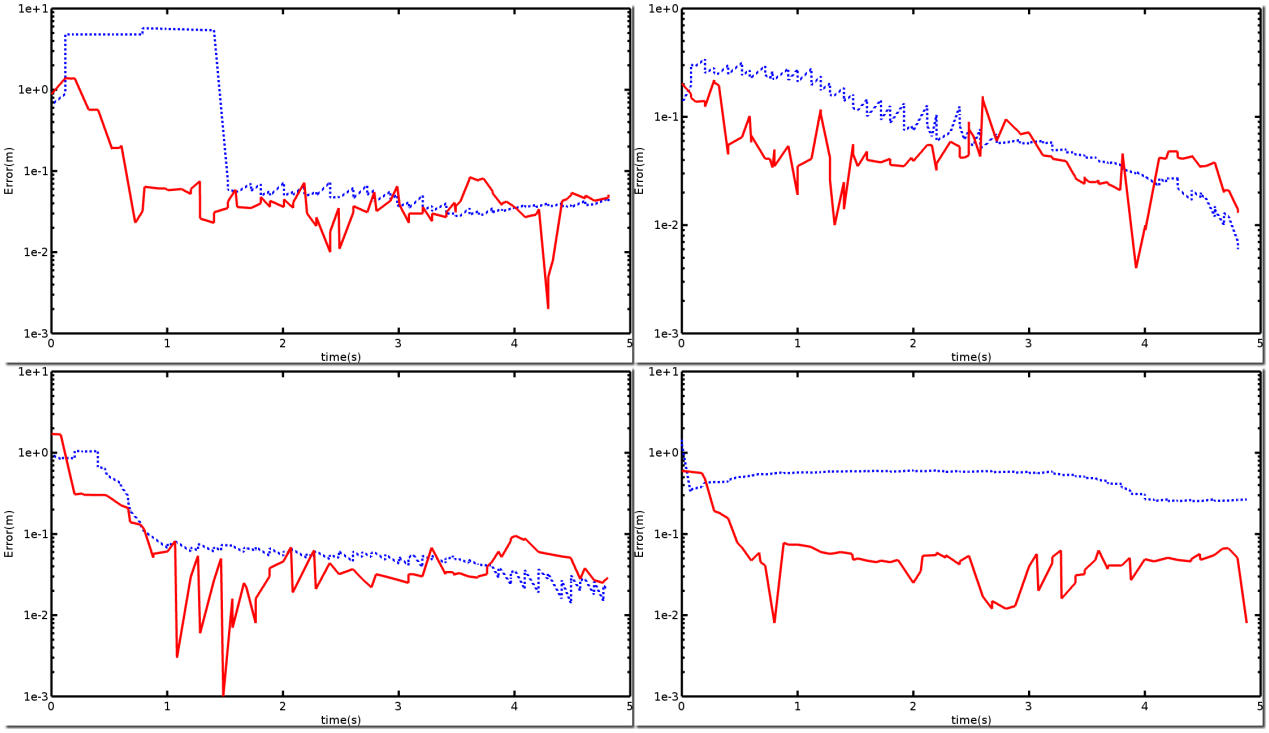

Fig. 5. Position estimate error in meters (y-axis) of PSGL (red solid line) and AMCL (blue dashed line) algorithms. AMCL is configured to a high update rate. The x-axis refer to the elapsed time, in seconds.

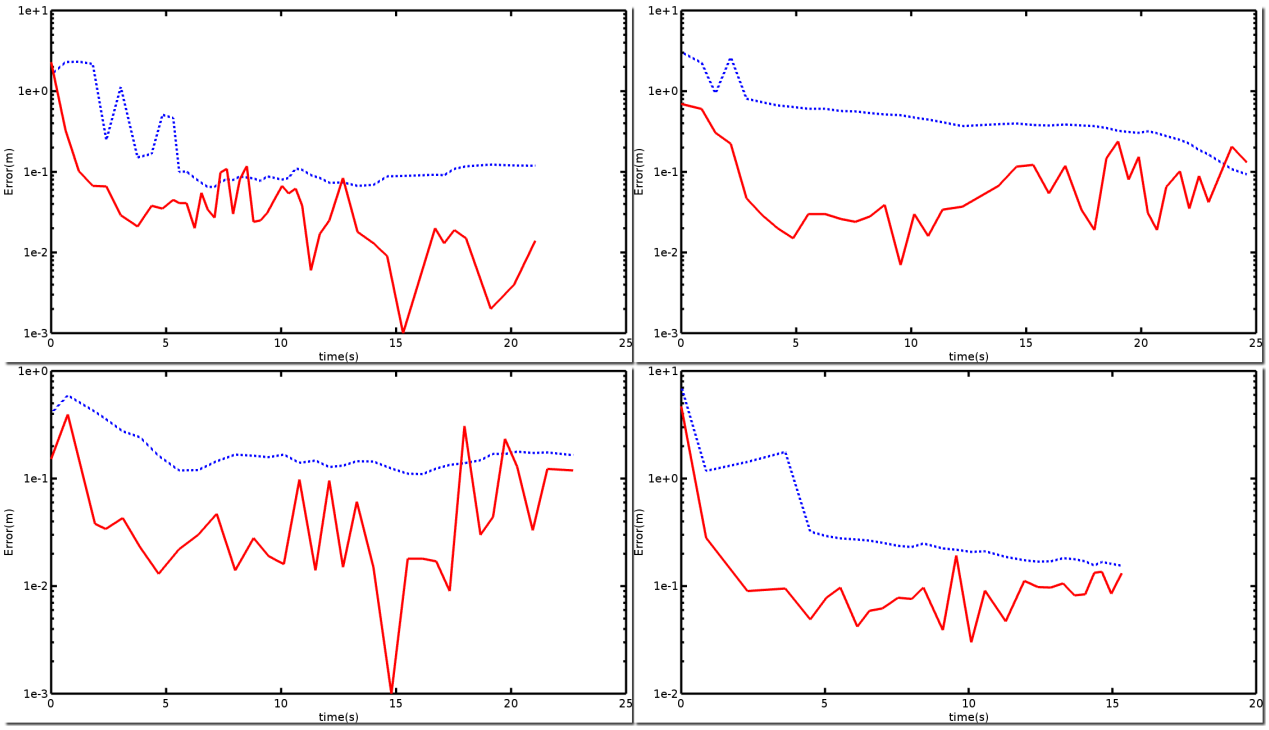

Fig. 6. Position estimate error in meters (y-axis) of PSGL (red solid line) and AMCL (blue dashed line) algorithms. AMCL is configured to its standard update rate. The x-axis refer to the elapsed time, in seconds.

TABLE I
EFFECTIVENESS OF PSGL BY POPULATION SIZE.

Population size	*Total of trials*	*Trials succeeded*	*Trials Failed*
10	40	9	31
15	39	13	26
25	41	24	17
50	40	21	19
100	42	28	14
200	42	33	9
400	42	37	5
600	42	37	5
800	45	41	4
1000	43	40	3

of particles should be proportional to the size of search space. The map of Fig. 1 has approximately 73 m^2 of free space, considering only the cells that can be visited. The Fig. 7 shows the relationship between the population size (x-axis) and the percentage of good estimates found (y-axis). In the experiments performed, PSGL produced poor performance when a density of less than 6 particles per square meter is used.

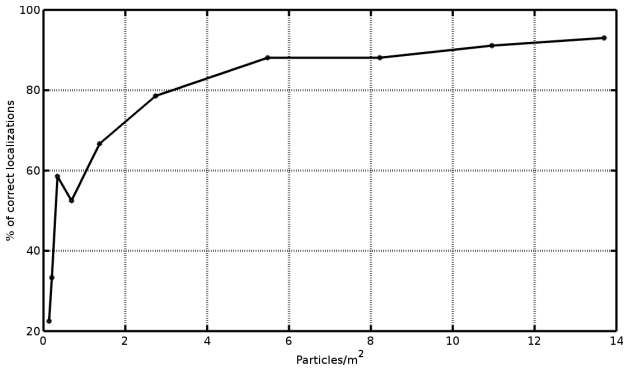

Fig. 7. Percentage of global localization good estimates $versus$ number of particles per m^2.

VI. CONCLUSIONS

This work presented a solution for a robot localization algorithm that integrates Pose Tracking and Particle Swarm techniques. The Global Localization technique of PSGL integrated with map matching approach of Perfect Match produced a robust localization method. The original PSO approach was incremented to update p_{best} particles according to the odometry information, keeping them synchronized with the vehicle movement, which is not addressed in [4], [6] and [7]. Results show that this approach can solve the problem of Global Localization with good performance, compared with the AMCL.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement N.777096 and from SEPIN/MCTI under the 4th Coordinated Call BR-EU in CIT.

REFERENCES

[1] P. Corke, *Robotics, vision and control: fundamental algorithms In MATLAB® second, completely revised*. Springer, 2017, vol. 118.
[2] M. Lauer, S. Lange, and M. Riedmiller, "Calculating the perfect match: an efficient and accurate approach for robot self-localization," in *Robot Soccer World Cup*. Springer, 2005, pp. 142–153.
[3] H. Sobreira, M. Pinto, A. P. Moreira, P. G. Costa, and J. Lima, "Robust robot localization based on the perfect match algorithm," in *CONTROLO'2014–Proceedings of the 11th Portuguese Conference on Automatic Control*. Springer, 2015, pp. 607–616.
[4] M. Pinto, A. P. Moreira, A. Matos, and H. Sobreira, "Novel 3d matching self-localisation algorithm," *International Journal of Advances in Engineering & Technology*, vol. 5, no. 1, p. 1, 2012.
[5] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proceedings of the IEEE international conference on neural networks*, vol. 4. Citeseer, 1995, pp. 1942–1948.
[6] Q. Zhang, P. Wang, P. Bao, and Z. Chen, "Mobile robot global localization using particle swarm optimization with a 2d range scan," in *Proceedings of the 2017 International Conference on Robotics and Artificial Intelligence*. ACM, 2017, pp. 105–109.
[7] A. R. Vahdat, N. NourAshrafoddin, and S. S. Ghidary, "Mobile robot global localization using differential evolution and particle swarm optimization," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 1527–1534.
[8] A. M. Pinto, A. P. Moreira, and P. G. Costa, "A localization method based on map-matching and particle swarm optimization," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 313–326, 2015.
[9] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, July 2000, pp. 84–88 vol.1.
[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
[11] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.